Combining Affective Intelligence with Learning to Improve Action Selection in Decision-Making Agents

Jason B. Williams
Department of Computer and Information Science
University of Massachusetts at Dartmouth
Xiaoqin Zhang
Department of Computer and Information Science
University of Massachusetts at Dartmouth

September 21, 2018

Abstract

Complex environments contain more information than either natural or artificial agents can fully process in a timely manner. Studies in neuroscience have demonstrated that natural agents utilize affect (or emotion) to filter out irrelevant inputs. In this work, we propose to integrate an affect filtering mechanism in artificial agents to improve the deliberation time for action selection in environment containing a massive number of selection options. To evaluate this model, we create two agent architectures: the first architecture is based on an active reinforcement learning algorithm and the second architecture utilizes a hybrid design with both active reinforcement learning and the affect-based filtering mechanism. We have compared the deliberation time and the overall utility score of these two agents in the same environment. The results showed that the affect-based filtering mechanism is effective in decreasing the deliberation time without compromising the agent's utility score. The results from this study strengthen the premise that affect plays an important role in intelligent behavior.

Keywords: Affective-Computing; Decision-Making; Agent Architecture; Machine Learning; Reinforcement learning

1 Introduction

Complex environments and sophisticate problems require the integration of different intelligent mechanisms to handle them efficiently. For example, swarm intelligence [1] observed in the behavior of ants, bees or fishes helps to solve distributed optimization problems, decision-making method borrowed from rational agents has been used to evolve the architecture of ANN (Artificial Neural Network) for any given problem [15], and soft computing has also been applied to design and develop intelligent autonomous robots [37]. In this paper, we present our work on incorporating an affective mechanism with reinforcement learning to improve agent's decision-making in a complex environment.

Affective Computing is the field of computer science that studies "computing that relates to, arises from, or deliberately influence emotions" [35] . Even though affective computing is a relatively new discipline, the study of emotion (or affect) and its impact on intelligence has been a part of the study of artificial intelligence since the beginning. For many researchers, however, the study of emotion focused on how it hindered the cognitive process. Many people naturally understood that too much emotion could adversely influence a person's problem solving ability. Some, however, understood the positive role affect had. Dr. Damasio

demonstrated that too little affect can also have an adverse effect on problem solving. "I had before my eyes the coolest, least emotional, intelligent human being one might imagine, and yet his practical reason was so impaired that it produced, in the wanderings of daily life, a succession of mistakes, a perpetual violation of what would be considered socially appropriate and personally advantageous" [13] . Simon, in 1967, postulated that artificial agents must have two main systems: one to act and another to monitor the environment as well monitor itself [39]. In his theory, the monitoring of self and environment falls under the purview of affect. In 2007, Baumesiter, et al. developed a theory of emotion being a feedback system. They theorized, like Simon before them, that emotions are not the direct cause of actions, but rather a retrospective appraisal of actions [7].

As more research is performed in the area of artificial intelligence and emotion, it becomes more evident that logic alone is not enough to make a truly intelligent artificial agent. As stated by Marvin Minsky, author of *The Society of Mind*, "the question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without any emotions" [29]. Dr. Damasio, an accomplished neuroscientist, performed a landmark study with human patients who were suffering from frontal lobe damage [13]. The frontal lobe is the area of the brain that controls and regulates emotion, especially higher level emotions that deal with cognition [32]. Dr. Damasio theorized that emotion serves as a biasing agent in problem solving [35], meaning that emotion acts as the stopping mechanism that signals the logical search mechanism to stop. Dr. Picard from MIT was the first to make a case for the study of affect to be given its own emphasis [35].

Given the limitation of resources and knowledge, and the computational intractability of most problems in real world applications, fully rational solutions are rare. The human system of emotions is a biologically based solution. It provides genetically based heuristics for situations that affect current action and that have recurred during evolution. It outlines scripts for coordination with others during cooperation, social threat, interpersonal conflict, etc.; and it serves as a base for constructing new parts of the cognitive system when older parts are found wrong or inadequate [48]. The ultimate objective of this research is to build a hybrid agent architecture, which integrates both rational reasoning mechanisms and emotional mechanisms. This project deals with the internal use of affect and how it might assist the decision-making process. The goal of this project was to create an affect model to answer the following question: *can affect be implemented in an artificial intelligent agent in order to decrease the deliberation time for action selection without adversely affecting the quality derived from the selected action*?

In general, there has always been a tradeoff between speed and accuracy. The more accurate the evaluation, the longer it will take. Ideally, an agent would have the time required to analyze every aspect of every choice presented to it before making a decision. This would ensure that it always made the best possible decision. However, most agents do not have this luxury. Agents, both natural and artificial, situated in complex environments do not have the time to evaluate every option. By complex environments, we mean environments which are not completely observable, which are dynamic, and which have a sufficiently large population of inputs that the agent must select from. By the time an agent in such an environment fully evaluated every option, the options would no longer be available.

If we assume that Dr. Damisio's patients with frontal lobe damage were like standard computer programs: having logical skills but no emotional ones, we could assume that given an unlimited amount of time, that Dr. Damasio's patients would have eventually explored all possible avenue and find the ideal solution. What his patients lacked was a way to terminate the logical evaluation. Essentially, they lacked a trigger to say "this is a good enough solution, stop looking."

In searching for an answer to the above stated question: can affect be used to speed up the action selection of an artificial system in such a way that does not diminish the quality of the selection, we are looking for an answer to the question: *can a computer*

system utilize affect to stop searching for a "good enough" solution has been found? And to answer that we must first ask, *what is good enough?* Since "good enough" is subjective and will change depending on the nature of the environment and the problem the agent is working to solve, we will not make a hard rule to determine if the results are good enough. Rather, we will simply look at the results achieved through a standard learning algorithm and compare those to the results from an affect-based threshold filtering algorithm and map the outcome. Someone wanting to use this methodology will then need to decide for themselves if it is worth it.

Real world environments, as opposed to game environments, are the most complex environments. There are too many inputs to be processed in a reasonable amount of time. This holds true even for some non-trivial game environments, such as the ancient strategy game Go. Competition games are played on a 19x19 grid, and there is only one action: place a stone on a grid intersection. Even with a game like Go that has a very simple game play structure can have a complex state space. How a master Go player chooses his actions is naturally a complicated process, and not one that will be tackled in this study. However, there is one item that can be studied: in complicated environments or complicated state spaces, most inputs are ignored. This permits the agent to focus on the subset of important inputs. In playing Go, it makes sense that the master player is not evaluating every possible move. She understands, through experience and the feel of the game, that there are moves that are not worth evaluating. But even Go, with its complex space, is still less complicated and therefore easier for an artificial agent to navigate, than real world environments.

How does the agent know which inputs are important enough to consider and which ones are not? That depends on the agent's current environment and its past experiences. Some inputs are more important in some environments than other. The agent will learn this through its own experience and the feedback it receives. The learning that the agent will do, will be loosely tied to the environment that it is in. But at the same time, the agent needs to be able to use that knowledge in other environments for it to be most useful. For a simple real world example, take the case of an agent wanting to get another agent's attention. We will assume that it has learned that calling that person or agent's name will get their attention. However, if this is done in a library, they will undoubted gain their objective, however there will also be negative consequences. We will assume also that the agent is socially conscious and does not want to earn the ire of everyone around it. The negative reaction he or she receives will teach them that in this environment, they should not use a loud voice to get someone's attention. However, it would be helpful if the agent could learn what it was about the library that made its action incorrect. If the agent were able to do this, then it could avoid a repeat of the negative reaction it received in other environments in places like movie theaters and restaurants.

The affect algorithms in this study will attempt to accomplish this. Rather than looking at specific objects and learning about them, the agent will learn to identify properties of objects and their impact on the reward or reaction they received. There will naturally be a learning curve for the agent, as it experiences with new environments and experiments with what properties lead to which rewards. Back to the example of calling out to someone in the library. If the agent associates the negative reward to the presence of books, it will have some success in other environments. If however, it learns to associate being quite with the presence of other quiet people, it will have greater success.

In a similar way that affect triggers the agent to stop looking for an action to select, namely when it has reached a given threshold of utility, affect also sets a threshold of importance for the agent. If the input is below the given attention threshold, it is not important enough to evaluate it at this time. Since this study deals with affect as thresholds, we will implement both filter thresholds: the good-enough filter (to stop searching) and the not-good-enough filter (to filter out unimportant). Between the two

thresholds, we will determine if this helps improve the action selection process in complex environments.

In the rest of this paper, we will discuss related work in Section 2. The design of the system is presented in Section 3. We will describe the details for the affective agent architecture in Section 4, and the other learning-based agent architecture for comparison in Section 5. Experiment results are reported in Section 6, and conclusion and future work are presented in Section 7.

2 Related Work

The interest of AI researchers was drawn to emotion by Damasio's declaration that rationality cannot be understood separately from emotion [13, 23]. A number of different computational models of emotion were proposed [10, 16]. The OCC model [33] based on cognitive appraisal theory has been widely adopted in many research projects, such as the affective reasoner [19] and the OZ project [6]. It describes the cognitive structure of emotions. It provides a systematic and detailed account of the cognitive generation of emotions. Instead of the discrete categories of emotions adopted in OCC model, some researchers used continuous dimensions [49] to describe emotion space. The two most common dimensions are "arousal" (calm/excited) and "valence" (negative/positive). The distinction of these two models are not insurmountable: continuous dimensions of emotions can be mapped into discrete emotion categories and discrete categories of emotions can be treated as regions in continuous space.

The research on emotion in the field of AI can be divided into several sub-areas. One area is to study "external emotion": how to express emotion and understand the emotion expressed by another agent/user [43], even across different languages [4]. The emphasis of this research direction, effect modeling, is on creating artificial agents that can interact with people in believable ways. This entails creating systems that can both detect emotions in others as well display appropriate emotional responses. There is a great need for this type of research as computers and artificial agents become more integral and ubiquitous in our lives. The applications of this research include computer games [50], personal assistant agents [12], pedagogical agents [24], embodied conversational agents [34], virtual reality [28], cyberbullying detection [51], and potentially many more [20, 47].

Another area is to study "internal emotion": how emotion affects decision-making and other rational processing. A strategic information system based on emotional agent is proposed [2] to bring emotion and flexibility into strategic decision making. Different technologies have been adopted in building autonomous agents with emotion. A computational framework [45] for emotion-based control was proposed that includes perception, attention, motivation regulation, action selection, learning, and motor control systems. Computational Architectures for Motivation, Affect and Learning (CAMAL) based on BDI (belief, desire, and intention) model is presented in [14], where affect values are considered in evaluating goal importances, belief status, association insistence and motivator intensity. Emotion is used in an agent model [46] to build an adequate and efficient response mechanism to relevant stimuli, and also match the current stimulus to past experience. FLAME (Fuzzy Logic Adaptive Model of Emotions) [18] was built using fuzzy rules to capture the fuzzy and complex nature of emotions. Similarly, fuzzy rules are extracted and used in a fuzzy classification of human emotions [11]. A rule-based filter program [38] was developed to qualify the agent's expression of its emotional state by its personality and social setting.

All the above work focuses on modeling emotions and building autonomous agents with emotion, while our work is more focused on studying the impact of emotion on the rational reasoning process. The following work is closely related to our project. Gratch has discussed how emotion model can contribute to classical planning methods in [22]. He suggested some high-level approaches such as using appraisal as search control, and altering the behavior of the planner according to the social relationships with other agents. However, there is no implementation or concrete ideas available yet. Belavkin has done some experimental work [8] to show that emotion modeling can improve the performance of problem solving because it provides the function of a

powerful heuristic method and hence is important for intelligence. This work is limited to some very simple problems. Scheutz and Logan [41] have built agents with both affective and deliberative control, they showed that such agents outperform other agents with only affective or deliberative capability in different types of environments. However, they did not explore deeply how these two types of capabilities can affect each other.

Learning is also an important technology in emotion research [40]. On one hand, there are many projects on apply learning techniques to detect, classify and generate emotions in the past two decades [3, 27, 31, 42]. PETEEI (a PET with Evolving Emotional Intelligence) [17] is a general model for simulating emotions in agents, which incorporates various learning mechanisms so that it can produce emotions according to its own experience. Four types of learning are modeled: learning about events, learning about users, learning about actions and learn about conditioning, where a specific object is associated with a certain motivational state or emotion. A modular hybrid neural network architecture (SHAME) [36] was developed for emotion learning, it learns from annotated data how the emotional state is generated and changes due to internal and external stimuli. A hybrid neural network is also used to classify social emotions [26]. On the other hand, much less work has been done in using emotion to facilitate learning by creating suitable conditions, which is this project is focused on. The most relevant work we can found are described here. Asynchronous learning by emotions and cognition (ALEC) [21] architecture provides an agent the capability to learn to make decisions in real world environments which takes into consideration two alternative adaption processes: emotional and cognitive. A connectionist architecture - Crossbar Adaptive Array (CAA) was developed [9], which uses emotion as a basic feature of its learning procedure, which is based on the understanding of emotion as internal desirability evaluation. None of above work has addressed the following issue - how agent can learn to adjust its emotional reaction so as to improve its performance, which is what we study in this project. We will adopt similar learning mechanisms like in above work, such as reinforcement learning.

3 System Design and Environment Setting

The focus of this work on studying the utility of affective mechanism inside rational agent. In Damasio's study with human subjects with affective disorders stemming from frontal lobe damage, one finding was that some patients could not come to a conclusion. They were capable of deductive reasoning, but were not capable of terminating their reasoning process and deciding on a choice. Dr. Damasio theorized from this that emotion served as a bias mechanism that stopped the searching process. Our hypothesis is that affect provides this biasing mechanism by supplying filtering thresholds to the agent. These thresholds control the agent's selection process. When the agent finds an object that gives it the reward that it expects, or wants, the agent selects that object and discontinues searching even before the agent has evaluated all possible input objects. At the same time, affect also serves as a threshold filter for inputs that are not currently relevant to the agent. This enables the agent to disregard inputs that have no bearing on the current search. Based on this hypothesis, we augment a learning algorithm enhanced with an affective filtering mechanism hoping to improve the efficiency of the agent's decision-making process in a complex environment.

The goal of this study not on developing new learning algorithm, since numerous good learning algorithms have already been devised to help artificial agents navigate through complex environments. Hence, we will not be creating a new learning or searching algorithm. Rather, we will be modifying an existing one, namely the active reinforcement learning algorithm [44], to utilize the affective mechanism that is created for this research. The results of this modified algorithm will be compared to the results of an unmodified implementation of the active reinforcement learning algorithm to determine that whether affect had a desirable influence. The implementation of this affective mechanism will be detailed in Section 4.

3.1 Active Agent and Affective Agent

Two new software agent architectures are created to test the hypothesis that affect, or emotion, works as a filtering mechanism in an action-selection process in a massive environment: a non-affect enabled action selection agent to serve as a control architecture (comparison baseline in experiment) and an affect enabled agent architecture to test the hypothesis. The non-affect enabled action selection architecture (the control architecture) implemented an active reinforcement learning algorithm to learn from the environment and learn to interact with it. It will be referred to as the *Active Agent* going forward. The experimental architecture, the affect enabled action selection agent, will explore the usefulness of an affect-based filtering system. It will be referred to as the *Affective Agent* going forward. The design of both architectures will be discussed in detail in Section 4 and 5.

Both active and affective agent will be situated, one at a time, in the same environment. They will be presented with a considerable number of input objects from which to choose. Each agent will evaluate the objects and select one that it expects to return the greatest reward. After the selection is made, the agent will then wait for a reward for its selection. The wait time is to simulate required work to complete the task associated with selecting the object. The wait time for each input object is determined randomly within bounded limits at the time the object is created, as part of the environment setting.

Once the agent receives the reward for its selection, it attempts to determine how each of the object's property impacted the object's overall reward. The agent will then use that knowledge in selecting future objects. However, in extremely large state spaces, the agent cannot rely on seeing the same object more than once. Since the objects are randomly generated, it is possible that duplicate objects will be encountered within the input stream, but this will be the exception rather than the norm. Agents will be frequently presented with unfamiliar input objects, therefore the agent will not be able to rely on the overall object reward to function in its environment. It must use the property impact on the object. This implies that the more properties the agent knows, the more accurate its predictions will be. Each agent therefore will want to employ a strategy that will help it learn new or unknown properties.

After each experimental run the agents will be graded based on two factors: the deliberation time, which is the time required to process the inputs and the overall reward earned. Both the speed of evaluating inputs and the overall reward are important factors in determine the value of this affective filtering mechanism. As seen in game systems with extremely large state spaces, games such as Go, to evaluate all possible moves to determine the optimal selection is time prohibitive. In addition, human players of Go are able to perform action selections with high utility in a short time frame. We propose that the affective filtering mechanism allows for this.

3.2 Environment Setting

To make fair comparison of the two agents, each environmental setting is fully specified by pre-generated input pulses, which determines the time length of this setting and the environmental input for agent at each pulse. One specified environmental setting last a certain number of pulses. In the beginning of each pulse, the agent is presented with an input file that consists of a minimum of 10,000 input objects. The reason for the large number of inputs is to model a real world environment where an agent cannot evaluate all inputs that it receives in a timely manner.

To demonstrate this, examine human's senses and the input they are constantly receiving. Most of that input we as people receive, we have learned to filter out when we don't need it. For example, look at a student sitting in a library reading a book. Even when the student's attention is focused on the book, her eyes are still receiving input from her peripheral vision. Assuming she is sitting by a window, her eyes will see the outside objects such as trees, people walking through campus, birds flying

overheard, etc. Her ears will continually receive audio input even in a quiet environment such as a library. People talking softly, pages being turned, someone coughing, etc. The same is true of her other senses as well. In addition to external input, she could also receive internal input from her own body. If she is hungry for example, she will feel hunger pangs. If she is a typical student, there is a good chance that she will be tired as well. These are all inputs that her brain will have to deal with. And depending on how long its been since she has eaten or slept, the intensity of the input will vary. With this system, we want to emulate a similar environment, one where the agent is presented a massive number of inputs to process. When reading a book, the agent's attention given to the current task is dependent on several factors: namely how important and interesting the content of the book is to her, and how strong other inputs are. If she has not been able to sleep for a day or more, the contents of the book does not have as high a reward value as sleeping.

After process the input file, the agent decides to select an object or not to select any. The agent is not required to select an object every pulse. In fact, it is anticipated that during the reward wait time after selecting an an object, the agent will reject most inputs. If the agent is switching to a new object frequently, it could indicate that there is a problem, such as the agent did not select an object of sufficient worth in the previous pulse.

The agent will receive a reward from selecting an input object. The value of the reward is determined by the environment. Each reward is awarded after a wait period. This is to simulate work needed to collect the object. Though in this study no real work is required for the agent because it is out of the scope. Our focus is selecting the object and learning from the reward received, rather than the work to collect the reward. However, the simulated work period is included to give the agent the opportunity to be committed to a specific object.

The wait period for the reward for selecting an input object is counted in pulses and is bound, for this study, between 3 and 8 pulses. While the agents are waiting for their rewards, they will continue to be presented with new input objects. This is to simulate that the agent will continue to receive input through its senses even while working on a task. The agents will need to decide if it should continue to wait for the current selection to complete its run or if it should stop and select a new object to act upon. If the agent selects another object, it will not receive the reward for the originally selected object because it does not accomplish the work. If the agent waits for the action to completion, it will receive the reward linked to the originally selected object.

The environment is responsible for generating new objects with determined reward for agent to choose from, at each pulse. The input objects that the agents will be collecting are designed to be generic containers which could represent any object in any environment. They are therefore named WorldObjects. The object is simply a container for attributes. Each attribute is associated with a value chosen from a set of possible values designated for this attribute. The example below shows an object with oval shape, red color and smooth texture:

$$[< shape, oval >, < color, red >, < texture, smooth >]$$

Each attribute-value pair (referred as property) has a certain number of reward points. The total of these reward points of all properties is the input object's reward. For example, given the reward points list below:

$$< shape, oval >= 11$$

$$< color, red >= 0$$

$$< texture, smooth >= 4$$

the object with with oval shape, red color and smooth texture brings a reward of 15 (11+0+4) points to the agent who collects it. The agent will never be given the number of reward points of an individual property. Its task is to infer this information through interacting with the input objects and determining how each property contributes to the object's reward.

The reward function, for this study, is a linear summation function based on each property. As will be discussed future in the section on the affective agent's design, the affective agent's reward system is not limited to linear rewards. However, the active agent does assume a linear reward function. So for this study, to be able to compare the results in a meaningful way, the rewards are limited to linear summation functions.

Each pulse begins when the main controller calls the agent's run method and ends after the agent has either selected an input object, or has decided not to select any input object. With this definition of a pulse, the length of each pulse will vary based on how long it takes an agent to evaluate the input and the input itself. A pulse will take longer for input objects with 20 attributes than with input objects that have 5 attributes. Also an input file with 50,000 objects will naturally take longer to process than a file with 10,000, which is the minimum object count for any run. With this variability, pulses will not be used to measure the time an agent requires to make its selection; rather the actual computation time for an agent to process the input file and make a selection decision is measured as the deliberation time. Pulses are for organizing the inputs and ensuring that each agent is presented with the same inputs. At the beginning of each pulse of the simulation, the environment will be refreshed with a minimum of 10,000 unique objects.

In the simulation experiments, an agent is been evaluated based on two factors, the reward it collected, and the actual running time to finish the experiment with fixed number of pulses. Next we will describe how each agent architecture is designed.

4 Affective Agent Architecture

The affective problem-solving agent architecture includes two levels: the affective (sub-cognitive) level and the cognitive level, as shown in Figure 1. Therefore, the evaluation process is broken into two steps. The first step happens in a low-level sub-cognitive evaluation of all objects, which is a quick scan of the object to see if it has value (elicits an emotional response). After this quick evaluation, the object is passed to the high level, the cognitive level evaluation. Since the first level evaluation happens outside of the normal cognitive process, it has been named the sub-cognitive process. The cognitive process likewise evaluates the input to determine if it has value (affect importance) to the agent. If it does, the agent determines what action to perform in relation to the affect associated with it. This design of determining affect and then determining an action to deal with the affect is in keeping with psychological studies that demonstrate this two-step process in natural agents [30].

When the agent receives a reward from the selected object after the waiting time, the affective learning process creates and modifies current affection-based rules, which are used by the affection-based evaluation process on the sub-cognitive level to filter a large number of input objects in the future.

Next we will describe the workflow of the system and each of these processes inside this affective agent in greater details.

4.1 Overview of the Work Flow

Figure 1 is an overview of the agent's internal working flow. First, the sub-cognitive process conducts an initial evaluation process of inputs, and produces a selected set input objects stored in a prioritized queue, *Attention List*. Then the control is passed to the cognitive process with the attention list by the sub-cognitive process. This cognitive process will evaluate the objects in the

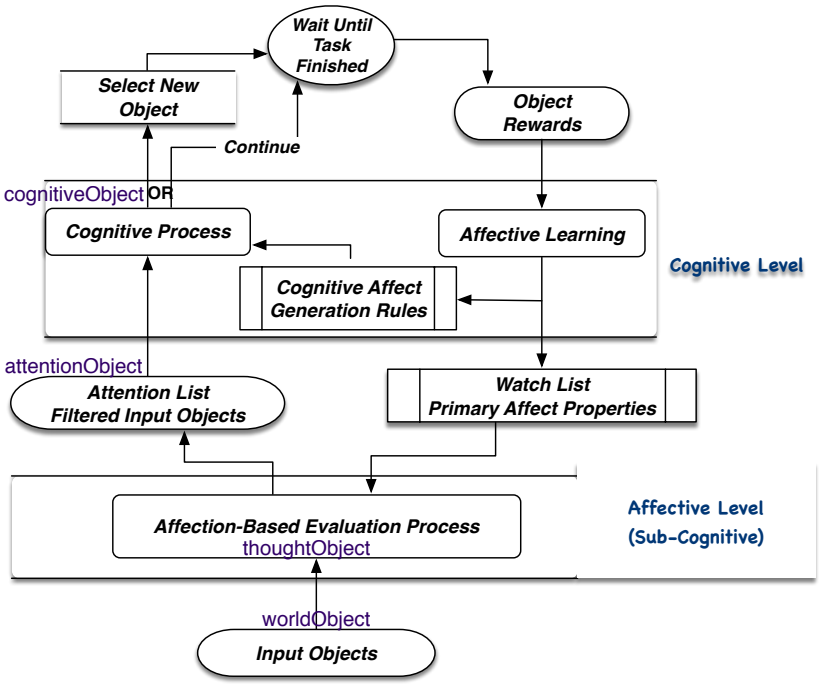


Figure 1: Affective Agent Architecture

attention list which require attention to determine if an action should be taken on any of the input objects *attentionObjects*. This cognitive process is a two-step process. The first step is to determine the input's cognitive affect and the second is to determine an action to handle the affect. Cognitive affect is a higher-level affect than the primary affect that is already determined by the sub-cognitive mechanism. Of these two steps, the first one, determining cognitive affect, is the focus of this study. Determining a relevant response for the affect is too broad of a topic to include in this study. For this study, the action is limited to selecting an input object from a large number of objects. The nature of this study is to determine the object with the highest reward level and select it. In future experiments, it might be instructive to add additional actions, such as to avoid selected objects, fight objects, teach objects, rescue objects, etc. The possibilities are limitless.

After this is completed, the cognitive process records the object that is selected and then control is returned to the main controller and the pulse is complete. The agent will then begin a new cycle starting once again with the sub-cognitive appraisal of a new set of WorldObjects. At the same time, the agent will be in an action simulation mode. The agent will be simulating performing the action (in our case, wait after selection) on the object. The agent will need to decide how much attention to place on its action and how much attention to place on evaluating new input objects. The agent will continue in this cycle of simulating work (by waiting) and evaluating objects while it waits for the reward from it action. When it receives that reward, the agent will learn about the object from the reward it received. This affective learning process learns how to produce two types of affects: primary affect and cognitive affect. Primary affect is used by the sub-cognitive process to filter new input objects. The knowledge of how to produce primary affect is represented as a *Watch List*, which is created and updated in this affective learning process and used by the affection-based evaluation process at the sub-cognitive level. Cognitive affect is used by the cognitive process to decide which new object to select and whether to interrupt the current work. The knowledge of how to produce cognitive affect is represented as a set of cognitive-affect-generation rules, which are created and updated by this affective learning process too. The agent then

looks for a new object to select using its updated knowledge at the beginning of the next pulse. This cycle will continue for the length of the experiment.

The length of the experiment is predetermined before it begins. For each pulse of the experiment, there is an associated input file. When the agent reaches the last pulse, it processes the last file. Once the file is processed, the session is finished. The agent will not receive any pending rewards that would come after that last pulse. After the run is finished, the total reward earned and the total running time elapsed are displayed and recorded.

At the end of the pulse, when control returns to the main controller of the system, if an object was selected by the agent, the reward for that object will be retrieved from a file. When each object is generated, its reward is also generated. The rewards are stored in a file that corresponds to the pulse where the object was presented to the agent. When the agent notifies the main controller that it selected an object, the controller retrieves that reward information.

Part of the reward information retrieved is the wait time for the reward. The wait time is added to simulate a time requirement to collect the object. In this simulation, the agent does not need to perform sub-tasks, since that would have no bearing on the aspect being studied. The agent will, however, need to wait for the reward to be added to the input stream if it wants that reward. If, during that wait period, the agent discovers an object with a higher reward, it can switch tasks to the new object, at which point the old object is forsaken and its reward is forfeited. This forces the agent to develop a commitment strategy for determining when to finish a task and when to abandon it for a more valuable object.

When the wait time has elapsed, the controller will add the reward object to the input stream. The reward object will then be read by the sub-cognitive process during the pulse where the wait time has passed. The sub-cognitive process will recognize it as a reward and process it accordingly. When the reward object is passed to the cognitiveSpace and processed, the agent is no longer working on that object that earns reward. This leaves the agent in a state of having no current action and ready to start a new action selection process without needing to compare the input to a current commitment.

4.2 Sub-Cognitive Process

We are constantly receiving input from our environment. Most of the time, we are unaware of it due to sub-cognitive filtering. In any environment having a reasonable degree of complexity, selecting an input object from among the available choices can be a time consuming activity. The more complex the environment, the more time is required to make the selection. At some point, the environment's complexity reaches a critical mass and many artificial agents are no longer able to make a selection within a reasonable amount of time. One solution to this problem is to build a system which reduces the number of inputs the agent needs to evaluate to make its selection. This method however raises the problem of how to reduce the number of inputs without first evaluating them. If the agent is not evaluating all of the inputs, how can we ensure that the agent is not discarding the best option?

This architecture's two-step process aims at answering this question. The first step in the solution is the sub-cognitive mechanism. The sub-cognitive process is separated from the primary cognitive functioning of the agent and, by definition, below the awareness of the agent. This is in line with studies on sub-cognition or sub-conscience mechanisms. This means that the sub-cognitive system is a sub-system of the agent, but one that is not within the agent's direct, or cognitive, control. It functions autonomously to prioritize the incoming objects. The cognitive mechanism, as we will see in the next section, give information to the sub-cognitive mechanism, but it does not control its actions. The two processes do communicate with each other, but neither process dictates actions to the other one. This separation of duties reduces the amount of processing the cognitive workspace has to perform on each input object, therefore allowing it to handle more complex environments in a shorter amount of time.

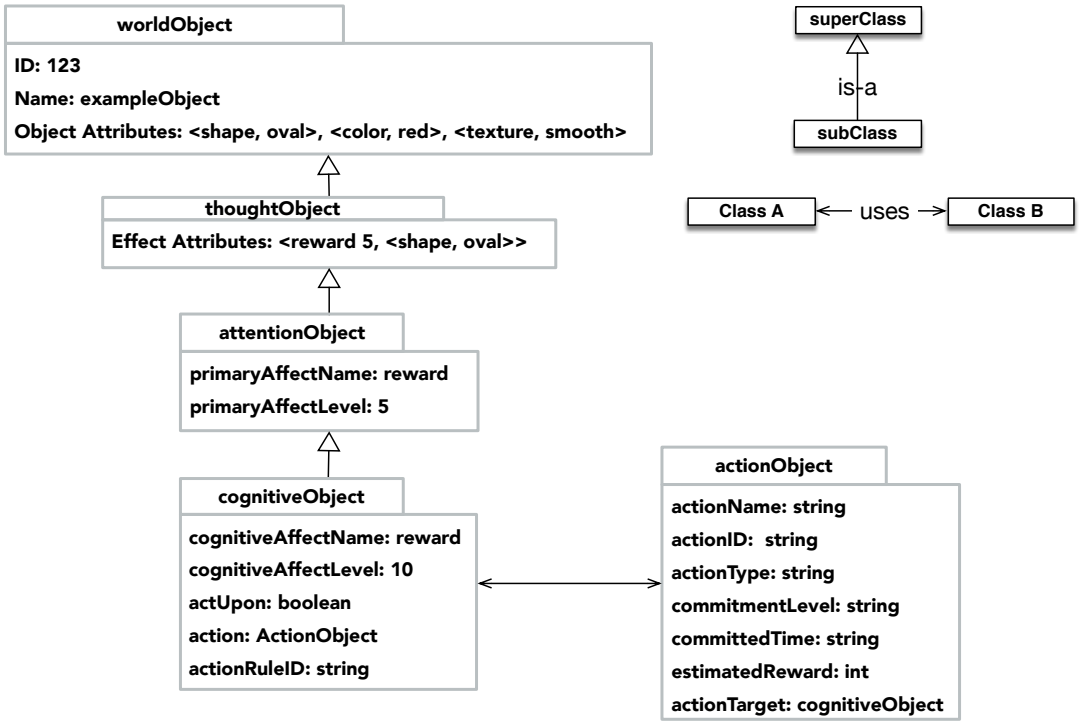


Figure 2: Different Object Types Used in Sub-Cognitive and Cognitive Process

This separation of control allows them to run in parallel if needed. For this study, however, the sub-cognitive and cognitive processes are run in series rather than in parallel so the run times would have a standard level of comparison with the active reinforcement agent.

This sub-cognitive mechanism performs a fast appraisal on all of the inputs. It looks at all inputs which prevents any from being skipped, but it does not perform a complete evaluation of any input. Instead it simply looks to see if the object has an important property. A complete evaluation entails determining the value of all attributes of the object, which will be the function of the cognitive process. In contrast, a fast appraisal examines the object for any important property. If it finds one, the process adds the object to its attention list and moves on to the next object. The objective of this first appraisal is not to find the value of the object. Rather it is to quickly determine if a object is worth spending the additional time to evaluate its value . The principle here is: if the object has at least one important property, the cognitive process should be made aware of it. Also, the more important a single property of the object is, the more likely that object is important to the agent. Therefore, the sub-cognitive level will present these objects to the cognitive level first.

There are two questions that need to be addressed. The first is how does the sub-cognitive process place a value on the property? And the second is how do we ensure that the most important objects are at the top of the queue? The agent maintains a list of properties that it is watching for. This *Watch List* is built as the agent learns about the objects in its environment. This will be discussed in greater detail in the Section 4.3 on the cognitive process. For now, it is sufficient to say that as the agent learns about its environment it starts to build a watch list. The watch list consists of single-attribute-value pair and its estimated reward: $\{A_i, P_i, W_i\}$. For example, an entry: $\{$*Shape, Oval, 10*$\}$ in this list means that when an object contains attribute *Shape* with value

as *Oval*, its estimated reward is 10. This list is maintained as a priority queue with the highest values at the top of the list. Every known property (attribute-value pair) is kept on this list. However, to make the list useful for prioritizing input objects, the watch list also has a threshold value t. We will explain how this threshold value t is set in Section 4.3.

When the agent processes the input file that containing at least 10,000 objects, each object is presented to the sub-cognitive level first. The input objects are presented to agent's sub-cognitive process in the format of worldObject, a data type that holds ID, Name and Object Attributes of an object. Figure 2 shows the different object data types used in the sub-cognitive and cognitive process. New data type is chosen when additional information is added to the original object. A thoughtObject is created to represent this worldObject, with space to store additional information Effect Attributes, which will be provided by the affect-based evaluation process at the sub-cognitive level. At this level of awareness, only the agent's sub-cognitive process has access to the thoughtObject.

As the sub-cognitive process performs an appraisal on each thoughtObject, it compares its attributes to the entries on this watch list. It starts at the top of the list with the attribute/parameter pair with the highest estimated reward. If the thoughtObject has a matched attribute/parameter pair, and the corresponding estimated reward is above the watch threshold t, then it triggers the generation of affection. As a result, an attentionObject (Figure 2) is created representing that this thoughtObject should be attended to the cognitive level.

Conceptually, the attentionObject represents input items that have passed though the sub-cognitive filtering process and been picked up by the cognitive process. The agent is now aware of it at a conscious or cognitive level. The attentionObject inherits all information from the thoughtObject, with two additional pieces of information: primaryAffect and primaryAffectLevel. The first one is the name of the affect generated, and the second one is an integer representing the strength of this affect, which is set as the estimated reward of the matched attribute/parameter pair. This attentionObject is added to the *Attention List* (a prioritized queue according to its primaryAffectLevel) and the scanning the remaining attributes of this input object is stopped then. Since the watch list is maintained in order of reward value, as the agent scans down through the list, it will naturally look for the most important parameters first. The parameters of the object that the search would find afterwards will be of lesser value, so there is no need to continue to scan the object. Inputs that do not have an attribute above the watch threshold, do not currently have any value to the agent. They will simply be added to the bottom of the attention list with a value of 0. They can be evaluated by the cognitive mechanism during idle time when it lacks anything of higher value to work on.

Naturally there will be situations where one input object will have several parameters with lower values that when added up will have a greater value than an input object having a single high-valued parameter. In this scenario, the second object will be placed higher in the attention list than the first object because the mechanism is not adding up the parameter values. This is to be expected and will be handled during the cognitive evaluation. It is in fact a good example of why the input objects require a full evaluation in the cognitive space before an input object is selected.

Since all known attribute parameter pairs are added to the watch list even if they have no value, the agent has a built-in mechanism to identify new attributes or new parameters of an existing attribute. As the agent appraises the input object, if it cannot find a parameter on the watch list, it will mark it as new and add it to the top of the queue. This will trigger the cognitive agent's curiosity mechanism. This allows the agent to learn about new parameters as they are introduced into the environment. However, this is not a fool proof way to identify new attributes or parameters. If the agent finds a high valued attribute, one whose estimated reward value is above the watch threshold value, before it discovers the new attribute or parameter, the agent will not

know to move it to the top. When this occurs, the agent has a second opportunity to discover the new attribute or parameter. Its cognitive process may catch it. It is also possible however that the cognitive mechanism may select another input object before it discovers the input object having the new value. This is not a flaw in the design. This mechanism mimics natural agent behavior. There are numerous situations where a person is so engaged in what they are working on, that they fail to notice something new in their environment.

4.3 Cognitive Process

The second level of affective filtering happens within the agent's cognitive process. The work done in the sub-cognitive process is to filter the inputs and to simplify this process. The cognitive process takes the top attentionObject from the attention list and determines if the agent even needs to be aware of it. This awareness is controlled by the attention threshold. The attention threshold is a measure of how much attention the agent is paying to the action it is working on, or how committed the agent is to the action. The more focused the agent is on the current action, the less it will notice other objects. The attention threshold is determined by the agent's expected reward for completing its current action. The more the agent expects to gain by completing the action, the more focused it will be on that action.

In fact, people filter out most inputs they receive when they are working on something. Other noises and sensory inputs fade to the background when a person is focused on reading a compelling book. Even internal inputs, such as hunger pains, will also fade to the background when a person is doing something engaging. In fact, the degree to which a person is able to ignore other inputs is related to the degree to which they are paying attention to their current activity. Most of the sensory input we receive does not get our attention, unless we have a condition such as attention deficit disorder (ADD). ADD is characterized by the inability of an individual to focus on a given task. Unrelated inputs draw their attention away from what they are working on.

Since the agent does not yet have cognitive affect for this new object, it uses the primary affect to determine if it is worth paying attention to. The agent compares the level of primary affect to the attention threshold. If the primary affect level of the new attentionObject is greater than the current action's attention threshold, the agent becomes aware of the new object. Otherwise, the agent ignores it and consider the the next object from the attention list.

Now that the agent is aware of this object, a new cognitiveObject (Figure 2) is created from the attentionObject pulled from the attention list. This cognitiveObject contains all information of the attentionObject plus additional information including cognitive affect, cognitive affect level, and other data to the new object. At this point, the agent is just evaluating this new object. The agent has not yet decided to discontinue working on its current object, it will do more evaluation on this object before it decides if it should stop its current action in favor of acting on this new input.

Once the agent is aware of the object, it conducts a two-pass process to select an object to act on. The first pass employs a production rule engine to generate the agent's cognitive affect for the object, described in Algorithm 1. A production engine is a system that provides rules for the agent to use to make decisions. These production rules take the form of If-then rules. The condition portion of the rule, the *if* portion or left hand side (lhs) of the rule, is tested against the current state of the system and when it matches, the action portion of the rule, the *then* portion or right hand side (rhs), is triggered. The rules for the engine can be as simple or as complex as needed. A general form of a rule in this study is:

$\{(A_1, P_1), ..., (A_k, P_k)\} \Longrightarrow R; (F)$

The lhs of a rule includes k ($k \geq 1$) attribute/parameter pairs, the rhs of the rule is the estimated reward value R. Each rule is also associated with a frequency strength F, which records how many times this rule has been verified by the collected reward from

the selected objects. Noted that each rule used here at cognitive level may consist of multiple attribute/parameter pairs, while each entry in the watch list at sub-cognitive level only contains one attribute/parameter pair associated with its estimated reward. This is to simulate that sub-cognitive level has a much simpler and quicker responding process than cognitive level.

For example, at cognitive level, a rule

$\{(Shape, Oval), (Color, Red)\} \Longrightarrow 5; (2)//strength$

means that an object with oval shape and red color brings reward of 5 units and this fact has been observed twice in the past.

Updating the rules of the engine is also one aspect of how the agent learns, which will be described in Section 4.4. As it has more experience with input objects it adds and modifies its rules to gain a better understanding of its environment.

In this engine, all rules are ordered according to the estimated reward R in decreasing order. New and updated rules are added to the priority queue according the estimated reward. One by one, the system state that the rule is compared to is the input object's attributes. If every attribute/parameter pair in the lhs of a rule has been found in an attentionObject, this rule is fired and the agent generates cognitive affect for this object. The cognitive affect level is set as R in the rhs of the fired rule. Algorithm 1 below describes this process.

Algorithm 1 Cognitive Affect Evaluation of attentionObject O

```
procedure COGNITIVE-AFFECT-EVALUATION(O)
    L: a list of all rules sorted by reward R in decreasing order;
    for each rule r in L do
        if r.lhs matches a subset of O's attribute/parameter pairs then
            O.cognitiveAffectLevel = r.rhs (r.R);
            break (for loop);
        end if
        mark rule r as checked for O;
    end for
end procedure
```

Many cognitive architectures utilize production rules to determine action. In system like Soar [25] and ACT-R [5], the agent needs to evaluate all of the inputs before making a decision. Some of these systems, ACT-R for example, do not enforce a complete match on the rule. They implement systems that allow for partial matches to enable the agent to work in partially-known environments.

The engine written for this architecture does require a complete match on the left hand side of the rule to fire that rule, but it does not require the agent to evaluate all inputs. This was done to enable the agent to make quicker decisions. Most often, biological agents do not take time to evaluate every possible action they could make. Once they determine an action that accomplishes a goal, they stop looking for further possible actions. However, this was not the case of Dr. Damasio's patients, who had front lobe damage. He recorded that many were unable to decide on an action even after making a logical case for the validity of the action. Without affect, or emotion, to signal that a decision is a good one, the agent would continue to evaluate all options. This seems like a preferred strategy in order to make the best decision, the agent would need to know all of the possible actions. However, in any non-trivial environment, evaluating all possible actions would take a prohibitively long time. So to emulate this, the production engine of this agent architecture makes the trade-off of best action for quick action. Evaluating the validity of this tradeoff is the nature of this study.

Once cognitive affect has been generated, the second process begins, which is, to select an action for this object. Right now, the only possible action for an object is to collect the object, so the second process is a process that sets the action to collect the object. In some future study, it would be straightforward to add a second production engine to the second phase of the process to handle more action options. A second phase engine would enable the agent to select an action from a list of possible choices to deal with the affect. It would also allow to agent to experience learning in this are as well. These new actions that it learns would be added to the engine as rules.

The affect evaluation process was separated into two passes to provide a layer of abstraction to the process. This allows the agent to have separate production rules to generate affect and rules to act on that affect when they are implemented. This allows a separation of concerns that permits more flexibility in adding more actions.

Once the object has passed through both processes, it is not guaranteed to have an action. It is possible that the agent determined that no action was necessary for the object. In that case, the agent will continue to the next input in the attention list. The attention threshold is only the first hurdle for the input object to overcome to be collected. After the complete cognitive affect evaluation and after the affect is assigned, the attentionObject is re-evaluated using Algorithm 2 to ensure that it is of higher value than the current input.

The sub-cognitive affect is generated based on only one attribute. The re-evaluation of an attentionObject considers all attributes in this object to determine the expected reward. The re-evaluation process is described in Algorithm 2. If the expect reward, which is the full cognitive affect after re-evaluation, is greater than the expected reward of the current action (waiting for completion of collecting object to receive the reward), the agent will abandon the current action and begin working on the new object.

Algorithm 2 Cognitive Affect Re-evaluation attentionObject O

```
procedure COGNITIVE-AFFECT-RE-EVALUATION(O)
    L: a list of all rules that have not been checked for O in Procedure 1 , sorted by reward R in decreasing order;
    for each rule r in L do
        if r.lhs matches a subset of O's unmatched attribute/parameter pairs then
            O.cognitiveAffectLevel += r.rhs (r.R);
            mark the matched attributes in O as matched;
            if there is no unmatched attributes left in O then
                break (for loop);
            end if
        end if
    end for
end procedure
```

4.4 Affective Learning

When the agent starts in a new environment, it starts with only one rule: curiosity. It is like a human baby in that it doesn't know things, but it is curious about everything. There is some randomness in this. The objects that it sees first are the ones that it will select first, since it knows nothing else. Then as it learns, it balances its curiosity with the affect that it already knows. The process of how this works is described in detail here.

The agent learns by evaluating the rewards it receives and then updating the expected reward for the rule that fired. Learning in the agent happens within the cognitive level since learning is a cognitive activity. The sub-cognitive mechanism learns only

as information is passed to it from the cognitive process. The cognitive mechanism does not control what happens in the sub-cognitive space, nor is it even aware of what is happening in that process, but it does pass information and directives to the process, and this is how the sub-cognitive process learns. The directives that the sub-cognitive process receives relate to adding specific items to the process's watch list.

As mentioned previously, a reward is received by the agent when it completes the task of collecting an object. This reward triggers the execution of learning process. When the learning process is executed, it first retrieves the input object that was acted upon and it retrieves the rule that fired to select the object in the Cognitive Affect Evaluation Process (Algorithm 1). The method first evaluates the rule to see if it was the curiosity rule. If the object was selected because it had unknown properties, the agent does not have a rule for some or all of these attributes. In this case, new entries will be created in the watch list of the sub-cognitive space and new rules will be created in the cognitive space. Otherwise, the existing rules will be modified.

The learning process is as the following:

1. Initially the agent has no knowledge about the environment, with only one rule: curiosity rule.
2. Create single-property rules, one for each attribute/parameter pair in the object earned reward. Since the agent does not as yet know how each property contributed to the reward, it will assign the entire reward to that property. Each of those rules will have a strength value of 1.
3. When encounter the same property: attribute/parameter again, update the existing rule that matches this property.
4. If the actual reward more than one standard deviation away from the expected reward, this rule is not accurate enough to predict reward, then it discard the current rule and create new rules containing additional attributes.
5. If the existing rule is considered accurate enough, its expected reward and strength will be updated as described in Equation 2. Whenever the agent finds that a rule is accurate in predicting the reward, it will strengthen that rule's strength value.

For example, the agent just received an actual reward $R1$ from an object with three properties:
$\{\langle A1, V1\rangle; \langle A2, V2\rangle; \langle A3, V3\rangle\}$
This is the first time these three properties are observed, so three single-property rules are created:
$P1 = \{\langle A1, V1\rangle\} = R1; (1)//strength$
$P2 = \{\langle A2, V2\rangle\} = R1; (1)//strength$
$P3 = \{\langle A3, V3\rangle\} = R1; (1)//strength$

Then, the agent received an actual reward $R2$ from an object with three properties:
$\{\langle A1, V1\rangle; \langle A2, V9\rangle; \langle A3, V3\rangle\}$
and this object was selected because of rule $P1$. Assume that $R1 = 3$, and $R2 = 5$, the standard deviation between 3 and 5 is 1.414, greater than 1, hence $P1$ is considered not accurate enough, so $P1$ is discarded and new rules $P4$ and $P5$ are created:
$P4 = \{\langle A1, V1\rangle; \langle A2, V9\rangle\} = R2; (1)//strength$
$P5 = \{\langle A1, V1\rangle; \langle A3, V3\rangle\} = R2; (1)//strength$
On the other hand, if $R1 = 3$, and $R2 = 4$, the standard deviation between 3 and 4 is 0.707, less than 1, hence $P1$ is considered accurate enough. The agent will keep $P1$ but modified it using Equation 1 and 2. The expected reward will be updated as $(3 * 1 + 4)/(1 + 1) = 3.5$ and the strength will be increased to 2.

When an existing rule is updated, the rule's output, or rhs, is updated, while the lhs conditions of the rule are not changed. The

rhs stores three values: the expected reward type, the expected reward value and a reward strength. The expected reward type is the name of the affect that the agent is expecting from collecting the object. The reward value is an average of previous reward values received for the given attribute combinations. The reward strength value is a count of how many times this rule has fired in reward learning process. Between the reward value and reward strength values, the agent can generate a weighted average with the new reward value it just received as the reward. This is done to keep an outlier from skewing the results. Assume that the current expect reward value is R, the reward strength is t, the newly observed reward is r, then the updated expected reward and the reward strength are as the following:

$$R = (R * t + r)/(t + 1) \tag{1}$$

$$t = t + 1 \tag{2}$$

For example, if the agent has received reward 200 (t) times as the result of a given rule, and the expected result for that rule is 100 (R) and then a new result comes in and the reward for this new object is 90 (R), we obviously do not want to simply average the two numbers together. We want to take the previous learning into account. In this example, during this learning step, the existing reward estimation would multiple the average ($R = 100$) by the strength value ($t = 200$) and generate an overall value of 20,000. This is the total reward received by that agent through this rule. Then the new value 90 is added and a total of 20,090 is calculated. This value is then divided by 201 ($t + 1$), which is the new count of times this rule has fired, and a new expected reward of 99.95 (updated R) and a new reward strength of 201 (updated t) is applied to the rhs of the rule.

The agent manages its rules through ordering them by priority. As the agent learns, the number of rules it knows grows. By ordering the rules by their expected reward, the agent is able to keep the most important rules at the forefront of its decision process. This is important so that the agent is not comparing the object to all of its rules. Once it finds a rule, it will fire it and move on to the next step. Since the rules are ordered, the agent can be assured that the expected reward in the current rule is greater than that from any successive rule.

To summarize, the learning process for the affective agent revolves around creating and editing rules in the production engine. As the agent gains more experience with input objects, by receiving rewards from collecting them, it refines its expectations by modifying and creating rules.

5 Active Agent Architecture

In order to determine the effectiveness of this affect-driven learning agent architecture, a comparison agent was created. This second agent will be placed in the same environments as the affect learning agent. The comparison agent for this experiment will employ the unmodified active reinforcement learning architecture. Both agents will then be scored on the total reward they receive as well as the running time to complete the trial. The scores and the times of both agents will be compared to determine how effective the affect agent's emotion mechanism is in comparison to a known architecture.

5.1 Active Agent's Work Process

The active reinforcement learning comparison agent will evaluate all of the attributes of every input object in determining the best action. It does not have a filtering mechanism like the affective agent. This is to provide a baseline for comparison with the affective agent.

The agent functions by selecting an object and evaluating the reward it receives for that object. This is the same as the affective agent. And like the affective agent, it is highly unlikely that it will see that same object again. Therefore, it will need to determine,

based on the overall reward and the reward value of each property (attribute/parameter pair) of the object. The agent will actively discover its environment by determining each property's reward value through multiple interactions it will have with that objects that have the given attribute.

At the start of each pulse, the agent retrieves the new set of worldObjects. The agent checks each object and evaluates its properties to determine which object would provide the best reward. The evaluation of an object is based on its attributes and their associated parameter values. The total expect reward of an object is the sum of all expected rewards for each of it's attribute/parameter pair, if its expected reward has been learned by this agent (See Section 5.2); otherwise, a curiosity value as expect reward for a unknown attribute/parameter pair. This curiosity value is the same for both the affective learning and active reinforcement learning agents. This ensures that both agents are equal in their dealing with unknown properties.

During the evaluation of all input objects, the agent only keep the object with the highest expected reward, and discard the objects with lower estimated reward. After evaluating all input objects, the agent will further consider the object with the highest expected reward. It compares the expected reward of this new object to that of the currently selected object. If the agent expects a greater reward for the new object, it will stop working on the current one and begin working on the new one. This means that the agent forfeit the reward from the current object it has been working on because it stops working on it before obtaining the reward. The agent does not know if the new object actually is a better selection, which is in line with most real-world situations. When a person stops working on one task to start on another one, they often will not know what the other reward would have been. They have their estimated value which they learned from previous experiences, but the actual reward is never known before received.

5.2 Active Reinforcement Learning

Learning in the active reinforcement agent happens when a reward is obtained after finishing working on an input object. This agent stores the past input objects selected and their attribute information. The current reward object is matched with the original selected input object, and its attribute information is retrieved. Since an object has multiple attribute/parameter pairs associated with it, and the agent does not know how each property contributes to the overall reward, it makes the assumption that each property is equally responsible for the reward value. The reward is therefore divided evenly between the attributes[1]. While this will be an inaccurate assumption for most inputs, it will still serve the purpose of allowing the agent to learn.

For example, a reward of 31 is obtained after collecting Object 1:

$\{\langle shape, oval\rangle; \langle color, red\rangle; \langle texture, smooth\rangle; \langle speed, slow\rangle; \langle attribute10, value\ 3\rangle\}$

then it is assumed that each attribute/parameter pair contributes a fifth portion of the reward: $31/5 = 6.2$. Now the agent has learned the expected reward for five attribute/parameter pairs:

$\langle shape, oval\rangle = 6.2; (1)//strength$

$\langle color, red\rangle = 6.2; (1)//strength$

$\langle texture, smooth\rangle = 6.2; (1)//strength$

$\langle speed, slow\rangle = 6.2; (1)//strength$

$\langle attribute10, value\ 3\rangle = 6.2; (1)//strength$

[1]Noted this initial reward split is different from the assumption of each property causes the whole reward in affective learning described in 4.4. The reason for this difference is that splitting reward among multiple properties is a rational choice, which is closer to the environmental setting where the reward function is a summation over multiple properties. On the other hand, assuming each property responsible for the whole reward is a rather simple premise that resembles the fact that affect response is rather quick and straightforward compared to rational reasoning.

Assume this is the first time the agent observed the above properties, each learned rule is assigned with a frequent strength value 1. The agent updates its knowledge of property rewards by first multiplying the reward value by the strength value to get a total reward from the average reward it stores. The new reward, the portion that each property receives from the object's reward, is added to the property's total value and its strength value is incremented. A new estimated reward is determined by then dividing the new total by the new strength value. This updating process is the same process that happens in the affective learning agent, described in Equation 1 and 2 .

Continuing example, a second reward of 20 is obtained after collecting Object 2:

$\{\langle \mathbf{shape}, \mathbf{oval}\rangle; \langle color, blue\rangle; \langle texture, granite\rangle; \langle \mathbf{attribute10}, \mathbf{value\ 3}\rangle; \langle attribute15, value\ 10\rangle\}$

then it is assumed that each attribute/parameter pair contributes a fifth portion of the reward: $20/5 = 4$. The expected rewards are updated for the two repeated attribute/parameter pairs are:

$\langle shape, oval\rangle = (6.2 + 4)/2 = 5.1; (2)//strength$

$\langle attribute10, value\ 3\rangle = (6.2 + 4) = 5.1; (2)//strength$

and the agent also learned the expected reward for the three new observed attribute/parameter pairs:

$\langle color, blue\rangle = 4; (1)//strength$

$\langle texture, granite\rangle = 4; (1)//strength$

$\langle attribute15, value\ 10\rangle = 4; (1)//strength$

The rest of active learning process is the same as the cognitive learning process in the affective agent. The main difference between these two agents is that the affective agent has an affect-based filter mechanism in its sub-cognitive level while the active agent does not have this mechanism.

6 Experiments

In order to test the hypothesis that that affect (emotion) works as an effective filtering mechanism for an action selection process in a massive environment, experiments with the affective agent and the active learning agent were conducted in groups or sets. Each experiment set consists of ten experiment runs. Each run in the set has the same number of pulses. For each pulse, there is an input object file, which contains an identical number of worldObjects. The variation in each run comes from the differences between worldObjects in each file. Each worldObject within a run is created by the worldObject generator with the same parameters.

The results from each experiment set would then be normalized to account for differences in inputs. This allowed for comparisons in agents between set parameters. A total of 13 sets including 130 runs of experiments have been performed.

6.1 Experiment Set 1: Baseline, No Learning, 1 Pulse, 10,000 Objects, Immediate Reward

The first experiment set was created to be a baseline set. It would be used to compare with the remained sets. As such, each run in this set consisted of a single input file having 10,000 worldObjects, and each object had 15 attributes. In all of the subsequent experimental runs, each input file will have a minimum of 10,000 objects, and some will have more. Since there is only one file, there is only one pulse, and the reward was immediately available. There is no reward waiting time, and there is no learning because there is only 1 pulse in each run. The reward for each object is determined when the object is generated. Each property has a reward value, and as that attribute and its parameter value are added to the worldObject, its reward value is added to the actual reward for the object. This sets a baseline of reward received from random selection of an object, since no learning has yet occurred.

	Affective Agent		Active Agent			
Set 1	Score	Time	Score	Time	Score Diff.	Time Diff.
101	179	484	193	467	(14)	17
102	190	463	112	467	78	(4)
103	167	526	97	481	70	45
104	179	445	129	528	50	(83)
105	184	563	132	475	52	88
106	205	567	122	497	83	70
107	177	502	151	550	26	(48)
108	190	446	139	452	51	(6)
109	164	466	133	472	31	(6)
110	175	428	125	615	50	(187)
Avg.	181.0	489.0	133.3	500.4	47.7	(11.4)
St. Dev	11.98	49.25	25.60	50.30	28.40	80.21

Table 1: Experiment Set 1: 1 pulse, 10,000 objects, immediate reward

Most importantly, this set of trials provided the reference run time for each agent to scan a single, large file and make a selection. Having this baseline will allow for comparisons of learning algorithms in following experiment sets. The results for this experiment set are shown in Table 1.

The experiments for this set were numbered 101 through 110. The average run time for the affective agent was 489 milliseconds and its average earned reward was 181. The standard deviations for both score and time are included in the table. The average run time for the active agent on the same run was 500 milliseconds with an earned average score of 133. The affective agent examined this baseline run 11 millisecond faster and earned 48 more points. The affective agent was 2.3% faster and scored 35.8% higher as compared to the active learning agent. Noted that none of them has learned anything or applied the learned knowledge since there is only one pulse. The reward difference and the tiny time difference (compared to the subsequent experiments) are mostly attributed to the randomness of the experiments.

6.2 Experiment Set 2: 300 Pulses, 10,000 Objects, Reward Waiting

The second set of experiments contained 10 runs of 300 pulses each. Both agents examine 300 pulses sequentially, one pulse at a time. Each of these 300 randomly generated input pulses contains 10,000 worldObjects, and each worldObject has 15 randomly generated attributes of the total 20 attributes available. This experiment set is the first set with a reward wait time. The reward wait times for each worldObject is set when the object is initially created, its reward value is randomly chosen between 3 and 8 pulses. This means that it would take between three and eight pulses before the agent learns the results of its actions. As mentioned previously, this wait time is to simulate the time required to complete a task to earn the reward. The results for this set are in Table 2, which shows that the affective agent outscored the active learning agent by 11% and completed its evaluation 30% faster.

Compare the affective agent's performance with the random selection baseline in set 1, where the affective agent has average reward as 181 and running time as 489 milliseconds for 1 pulse. With 300 pulses in Set 2 and delay reward time from 3 to 8 (average 5.5), the agent receives reward every 5.5 pulses ignoring the possibility that agent may forfeit the reward of a chosen object by selecting another object during the waiting period. The number of selected objects during the 300 pulses is about

	Affective Agent		Active Agent			
Set 2	Score	Time	Score	Time	Score Diff	Time Diff
111	13,459	78,823	11,959	106,706	1,500	(27,883)
112	12,179	77,482	11,173	97,320	1,006	(19,838)
113	12,836	80,264	11,425	90,345	1,411	(10,081)
114	13,182	75,482	12,158	99,119	1,024	(23,637)
115	12,475	72,918	11,052	99,368	1,423	(26,450)
116	13,623	74,022	11,894	102,373	1,729	(28,351)
117	13,766	74,723	10,904	91,760	2,862	(17,037)
118	12,400	75,951	12,193	96,214	207	(20,263)
119	11,806	74,283	10,461	102,622	1,345	(28,339)
120	12,133	73,427	11,967	98,335	166	(24,908)
Avg.	12,785.9	75,737.5	11,518.6	98,416.2	1,267.3	(22,678.7)
St. Dev	689.14	2,416.13	600.16	4,932.58	769.26	5,924.01

Table 2: Experiment Set 2: 300 pulses, 10,000 objects/pulse, 15 attributes, reward after waiting time

	Affective Agent		Active Agent			
Set 3	Score	Time	Score	Time	Score Diff	Time Diff
121	33,252	157,031	28,636	187,305	4,616	(30,274)
122	35,216	153,486	30,205	187,820	5,011	(34,334)
123	30,348	147,722	25,191	185,255	5,157	(37,533)
124	35,641	155,009	29,576	192,103	6,065	(37,094)
125	34,360	152,493	30,721	198,689	3,639	(46,196)
126	32,842	154,518	28,217	188,828	4,625	(34,310)
127	32,409	157,886	27,856	195,844	4,553	(37,958)
128	33,100	156,066	29,822	202,146	3,278	(46,080)
129	33,165	155,715	28,456	201,616	4,709	(45,901)
130	33,411	159,952	31,126	201,364	2,285	(41,412)
Avg.	33,374.4	154,987.8	28,980.6	194,097.0	4,393.8	(39,109.2)
St. Dev	1,491.10	3,340.01	1,725.50	6,614.13	1,066.96	5,597.74

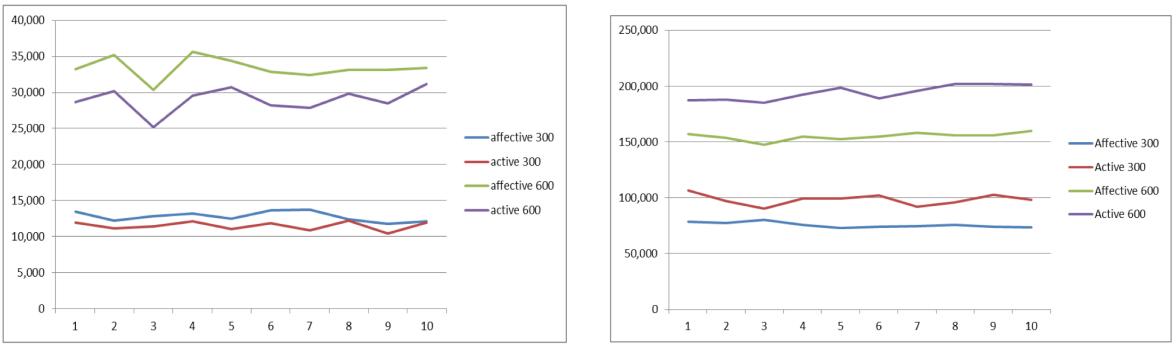
Table 3: Experiment Set 3: 600 pulses, 10,000 objects/pulse, 15 attributes, reward after waiting time

(300-5)/5.5= 54. The total reward of randomly selection is 54*181 = 9774. The affective agent with learning achieve reward of 12,785, which is about 31% more than random selection. The affective agent's running time is 75,737 milliseconds for 300 pulses, which is 252 milliseconds per pulse, compared to the random selection baseline 489 milliseconds for 1 pulse, there is no obvious overhead caused by learning.

6.3 Experiment Set 3: 600 Pulses, 10,000 Objects, Reward Waiting

Most of the parameters for experiment Set 3 are the same as those of Set 2. The one parameter that is changed for this set is the number of pulses per experiment run. Each experiment in Set 3 has 600 pulses, which is doubled from 300 in Set 2. This was done to give insight into the learning mechanism of each agent over time. Would the agents converge in reward or time the longer the experiment was run? Each file was randomly generated, so the agents would continue to see mostly new worldObjects. The increased duration would give the learning mechanism more time to converge on the actual reward values for the properties. The results of the experiment are in Table 3.

Table 3 shows that the affective learning agent outscored the active agent by an average of 4,393 points (15.2%). The affective



(a) Score Comparisons

(b) Time Comparisons

Figure 3: Score and Time Comparisons for Set 2 (300 pulses) and Set 3 (600 pulses)

agent also completed its evaluations quicker, by an average of 39,109 milliseconds (20.1%). Comparing the results of this set to the previous set, we see that the affective agent increased its score from 12,785 for 300 pulses to 33,374 for 600 pulses, an increase of 161%. We also see that the active learning agent increased from 11,518 points to 28,980 points an increase of 151.6%. The affective agent's time increased between sets from 75,737 milliseconds to 154,987 milliseconds, an increase of 104.6%. The active agent's time increased from 98,476 to 194,097, an increase of 97.1 %.

Since the number of pulses doubled, a linear increase in time to completion would be expected. Both agents did close to that. The affective agent was 4.6% over the 100% time increase and the active agent was 3% under it. Also, if the agents had had a linear increase in rewards earned, it could be assumed that the agents had learned all they could about the environment in the first 300 pulses, and they were making the best choices possible. However, since both agents improved on their reward score by more than 100%, it would indicate that they are both still learning, and as they have more experience in the environment, they are making better selections.

The two graphs in Figure 3 compare the results of Set 2 and 3 in terms of rewards earned and completion time. As Figure 3(a) shows, the rewards are comparable. With 300 pulses in Set 2 experiments, the scores of the affective learning agent and the active agent are more closely aligned. The gaps between their scores are greater with the 600 pulses in Set 3. Similarly, in Figure 3(b), when the number of pulses is doubled from Set 2 to Set 3, the time gap between these two agents increased: the affective learning agent saves more time.

6.4 Experiment Set 4 and 5: 900 / 1200 Pulses, 10,000 Objects, 15 Attributes, Reward Waiting

Experiment Set 4 has the following settings: 10 experiments, each experiment runs for 900 pulses. There is one input file for each pulse, which contains 10,000 objects. Each object has 15 random attributes, and a random reward delay between 3 and 8 pulses. This set has the same file and object settings as sets 2 and 3. The file count, and therefore run duration, is increased to 900. The results are displayed in Table 4.

The affective learning agent outscored the active learning agent by an average of 3,770 points (10.5%) and completed the set an average 57,969 milliseconds faster (23.6%). The affective learning agent experienced an 208.9% increase in reward earned in moving from 300 to 900 pulses (12,785 points to 39,504 points). The agent experienced a 18% increase in earned rewards from 600 pulses to 900 pulses (33,374 points to 39,504 points). Unsurprisingly, the longer the agent is in an environment, the more it

	Affective Agent		Active Agent			
Set 4	Score	Time	Score	Time	Score Diff	Time Diff
131	39,632	245,142	34,862	304,819	4,770	(59,677)
132	39,232	248,417	33,849	293,060	5,383	(44,643)
133	40,116	242,023	37,625	301,926	2,491	(59,903)
134	40,731	247,292	36,658	287,199	4,073	(39,907)
135	39,296	250,484	35,832	308,938	3,464	(58,454)
136	39,564	261,953	34,423	305,938	5,141	(43,985)
137	38,361	253,092	35,577	318,119	2,784	(65,027)
138	37,936	238,609	37,680	314,221	256	(75,612)
139	41,258	247,812	35,659	315,759	5,599	(67,947)
140	38,920	216,803	35,176	281,345	3,744	(64,542)
Avg.	39,504.6	245,162.7	35,734.1	303,132.4	3,770.5	(57,969.7)
St. Dev	1,011.05	11,796.09	1,273.67	12,395.70	1,631.94	11,595.59

Table 4: Experiment Set 4: 900 pulses, 10,000 objects/pulse, 15 attributes, reward after waiting time

knows about the environment, and the better it performs.

The active agent experienced a 23% increase in reward moving from 600 pulses to 900 pulses, over double that of the affective agent. The active agent also experienced a 210% increase in reward points moving from 300 to 900 pulses. Whereas the active agent improved twice as much as the affective agent between 600 and 900 pulses, the difference between the agents during the increase from 300 to 900 was only a 2% difference in favor of the active learning agent. This would seem to indicate that the affective learning agent's reward estimation converges quicker to the actual values, but by 900 pulses, they had both converged.

The affective agent experienced a 58.2% increase in completion time between 600 pulses and 900 pulses, and a 223.7% increase in completion time between 300 pulses and 900 pulses. The active agent experienced 56.2% increase in completion time between sets 3 and 4, and a 208.0% increase between sets 2 and 4. The active agent had a smaller increase percentage in completion times with the increase in pulses. However, the affective agent completed quicker than the active agent in all three sets.

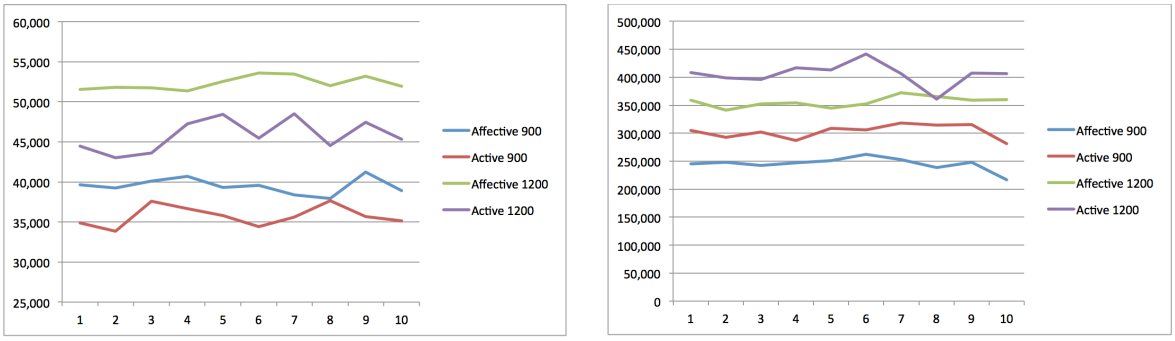
Experiment Set 5 increased the length of each experiment run to 1200 pulses, and retain other setting the same as Set 4. Table 5 shows the result of Set 5. The affective agent outscored the active reinforcement agent by an average of 6,528 points (14.2%). It also completed its evaluations an average 49,271 milliseconds faster (12.2%). Between the 900 pulses and the 1,200 pulses, the affective agent has a 32.4% increase in reward, and the active agent earns a 28.1% increase.

We noted that in one of the runs, number 148, the active learning agent completed the run quicker than the affective learning agent. This was different from the other nine runs in the set. In all ten runs, as is the case with all of the experiment sets, the data is randomly generated using the same parameters. In the other nine runs, besides run 148, the affective agent completed the between 33 seconds and 89 seconds faster. However, in run 148 the active agent completed the course 5 seconds faster. It is not clear what caused this outlier. With the complexity and size of the randomly generated data, it is not possible to determine its cause with any certainty. But, as is the case with randomly generated data, outliers do happen. This one instance does not significantly alter the results obtained from the entire set of experiment runs.

Figure 4 compares the results of Set 4 (900 pulses) and 5 (1200 pulses) in terms of rewards earned and completion time. Both affective agent and active agent have increased in reward and in time. Figure 4(a) shows the average scores by agent as the number of pulses increases from 900 to 1200. The graph shows the effect of learning on the agents. As the number of performance pulses increases, the affective agent scores better, indicating that its learning mechanism is having a positive effect on the selection

	Affective Agent		Active Agent			
Set 5	Score	Time	Score	Time	Score Diff	Time Diff
141	51,557	359,158	44,482	408,000	7,075	(48,842)
142	51,821	341,191	43,023	398,382	8,798	(57,191)
143	51,749	352,813	43,611	396,232	8,138	(43,419)
144	51,367	354,339	47,234	416,635	4,133	(62,296)
145	52,573	344,890	48,420	413,137	4,153	(68,247)
146	53,602	352,510	45,440	441,669	8,162	(89,159)
147	53,475	372,191	48,478	406,145	4,997	(33,954)
148	52,022	365,967	44,558	360,602	7,464	5,365
149	53,215	358,717	47,473	407,289	5,742	(48,572)
150	51,954	359,692	45,328	406,095	6,626	(46,403)
Avg.	52,333.5	356,146.8	45,804.7	405,418.6	6,528.8	(49,271.8)
St. Dev	825.22	9,189.94	1,972.78	20,146.84	1,697.65	24,599.83

Table 5: Experiment Set 5: 1200 pulses, 10,000 objects/pulse, 15 attributes, reward after waiting time



(a) Score Comparisons

(b) Time Comparisons

Figure 4: Score and Time Comparisons for Set 4 (900 pulses) and Set 5 (1200 pulses)

process. Figure 4(b) shows the average completion time by agent fas the number of pulses increases from 900 to 1200. The completion time increase for both agents is roughly linear as the pulse increases from 900 to 1200. The filtering mechanism of the affective agent consistently reduces the deliberation time for the agent over the unmodified learning algorithm used by the active agent.

6.5 Experiment Set 6 and 7: 300 / 600 Pulses, Double Number of Objects, Reward Waiting

For experiment Set 6, the number of worldObjects in each pulse is doubled over the number of objects per pulse in the previous sets to 20,000. The other parameters for this set of trials remain the same as Set 2: 300 pulses per run, 15 random attributes per object, and a random reward delay between 3 and 8 pulses.

The results from Set 6 are shown in Table 6. The affective learning agent outscored the active reward agent by an average of 1,346 points (8.9%). The affective agent outperformed the active agent on completion time by an average of 25,626 milliseconds (21.1%). The reward earned by both agents was comparable, but the affective agent was able to perform 21% faster.

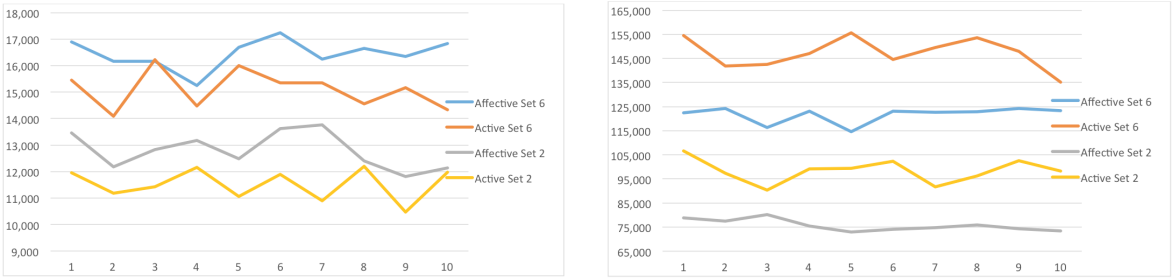
Comparing Set 6 to Set 2, the results in Figure 5 show a 28% increase in reward for the affective agent between these sets and a 31% increase in reward for the active agent between these sets. The active agent had a larger improvement on the 20,000 object

	Affective Agent		Active Agent			
Set 6	Score	Time	Score	Time	Score Diff	Time Diff
151	16,890	122,328	15,445	154,508	1,445	(32,180)
152	16,161	124,163	14,082	141,932	2,079	(17,769)
153	16,161	116,307	16,218	142,644	(57)	(26,337)
154	15,237	123,097	14,478	147,156	759	(24,059)
155	16,685	114,607	15,991	155,700	694	(41,093)
156	17,236	123,028	15,355	144,648	1,881	(21,620)
157	16,249	122,551	15,354	149,527	895	(26,976)
158	16,650	122,796	14,551	153,647	2,099	(30,851)
159	16,354	124,297	15,175	147,857	1,179	(23,560)
160	16,827	123,346	14,339	135,161	2,488	(11,815)
Avg.	16,445.0	121,652.0	15,098.8	147,278.0	1,346.2	(25,626.0)
St. Dev	551.42	3,348.75	715.05	6,437.46	794.00	8,076.44

Table 6: Experiment Set 6: 300 pulses, 20000 objects/pulse, 15 attributes, reward after waiting time

	Affective Agent		Active Agent			
Set 7	Score	Time	Score	Time	Score Diff	Time Diff
161	35,073	236,964	31,890	274,894	3,183	(37,930)
162	36,057	239,036	33,013	306,521	3,044	(67,485)
163	34,702	251,579	31,851	304,696	2,851	(53,117)
164	35,155	253,294	31,567	304,913	3,588	(51,619)
165	35,690	264,588	31,677	312,903	4,013	(48,315)
166	34,162	251,560	33,019	305,010	1,143	(53,450)
167	35,570	252,392	32,325	300,753	3,245	(48,361)
168	35,073	256,921	32,864	306,133	2,209	(49,212)
169	35,010	227,099	31,223	287,668	3,787	(60,569)
170	37,593	237,290	32,599	310,418	4,994	(73,128)
Avg.	35,408.5	247,072.3	32,202.8	301,390.9	3,205.7	(54,318.6)
St. Dev	930.89	11,414.62	649.27	11,499.08	1,040.08	10,230.14

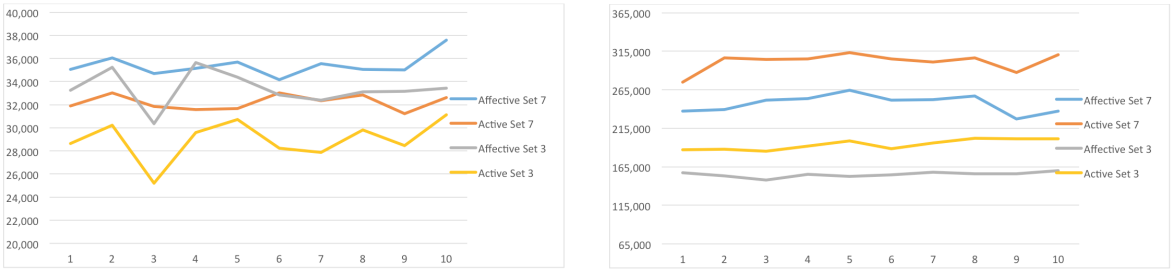
Table 7: Experiment Set 7: 600 pulses, 20000 objects, 15 attributes, delay reward



(a) Score Comparisons

(b) Time Comparisons

Figure 5: Score and Time Comparisons for Set 2 (10000 objects/pulse) and Set 6 (20000 objects/pulse), both 300 pulses



(a) Score Comparisons

(b) Time Comparisons

Figure 6: Score and Time Comparisons for Set 3 (10000 objects/pulse) and Set 7 (20000 objects/pulse), both 600 pulses

per pulse than the affective agent did, however, the improved performance is insufficient. The affective agent still outperforms the active agent in both reward and completion time.

Experiment Set 7 doubled the number of pulses over Set 6 and the other settings remained the same as Set 6. Set 7 uses the following settings: 600 pulses per run, 20,000 world objects per pulse, 15 random attributes per object, and a reward wait time between 3 and 8. The results of the experiment are shown in Table 7. The affective learning agent outscored the active learning agent by an average reward of 3,205 points (9.9%) and outran it by an average time of 54,318 milliseconds (18.0%).

Comparing Set 7 to Set 3, which differed only by the number of objects per pulse, Set 3 has 10,000 objects each pulse while Set 7 doubles it. Figure 6 shows a 6% increase in reward for the affective agent and an 11% increase for the active agent. Noted that the increase of reward is not expected because that the number of objects per pulse doubles, since the agent can only select at most one object each pulse no matter how many objects are available at each pulse. However, the increase of deliberation time is expected because more input objects need to be considered at each pulse. The affective agent experienced a 59% increase in completion time while the active agent had a 55% increase in completion time.

Comparing Set 7 to Set 6, where the difference was in the number of pulses in the run , the affective agent had a 115% increase in reward and the active agent had a 113.2% increase in reward. The affective agent had a 103.1% increase in completion time, and the active agent had a 104.6% increase. The learning increase was consistent across both agents with the increase from 10,000 objects to 20,000 objects.

6.6 Experiment Set 8, 9 and 10: 100 Pulses, 10000/10000/50000 Objects, 15/10/10 Attributes, Reward Waiting

Experiment Set 8 had the following settings: 100 pulses per run, 10,000 world objects per pulse, 15 random attributes per object, and a reward delay between 3 and 8 pulses. The results are shown in Table 8. The affective agent outscored the active agent by an average score of 206 points (5.4%). The affective agent ran quicker than the active agent by an average time of 7,213 milliseconds (21.5%). With only 100 pulses in the run, the earned rewards are distributed in a greater range of values compared to other experiment sets with longer duration.

In experiment 171, the active agent outscored the affective agent by 204 points. In experiment 176 however, the affective agent outscored the active agent by 892 points. This range of scores demonstrates that with shorter durations, the nature of the objects is more important than the composition of the properties of the worldObject. As the agent learns an expected reward value for an object's properties, and as it sees those properties again, those values influence the selection process. With only 100 runs, the agents do not have the opportunity to normalize their estimated values. This means that they could be quite inaccurate depending

	Affective Agent		Active Agent			
Set 8	Score	Time	Score	Time	Score Diff	Time Diff
171	3,936	24,210	4,140	32,143	(204)	(7,933)
172	4,120	27,106	4,092	32,045	28	(4,939)
173	3,730	26,552	3,626	33,274	104	(6,722)
174	4,095	27,499	4,077	35,064	18	(7,565)
175	3,936	26,667	3,794	33,128	142	(6,461)
176	4,489	26,879	3,597	33,539	892	(6,660)
177	3,697	25,963	3,547	31,326	150	(5,363)
178	4,176	26,257	3,808	34,239	368	(7,982)
179	3,827	26,550	3,238	33,830	589	(7,280)
180	3,998	26,017	4,021	37,251	(23)	(11,234)
Avg.	4,000.4	26,370.0	3,794.0	33,583.9	206.4	(7,213.9)
St. Dev	234.84	894.29	293.92	1,698.90	324.68	1,735.66

Table 8: Experiment Set 8: 100 pulses, 10000 objects/pulse, 15 attributes, delay reward

on the other attributes of the worldObject the agent selected.

For example, for the affective agent, if the reward returned for an object selected was high due to one or two properties, the rules the agent creates will attribute that reward value to all of the object's properties. When a second object is rewarded, those rules will most likely be abandoned for new, more accurate rules. However, with a small duration, fewer rules will have be normalized to find the accurate attribute parameter value. The same is true for the active agent. The reward portion that it distributes to each property will be an average of all of the properties' actual reward values. Depending on which properties were seen first, one or two properties could skew the values for the rest of the properties. Naturally, the longer the agents are able to work within an environment, the better each one is at predicting the results of its actions.

Experiment Set 9 also has 100 pulses per run, 10,000 objects per pulse, but only 10 random attributes per world object with a reward delay between 3 and 8. The difference between Set 8 and Set 9 was only the number of random attributes assigned to each worldObject. Set 8 had 15 while Set 9 had 10. This was done to see what effect the number of attributes had on the agent's learning. Both Set 8 and Set 9 serve as a baseline comparison for Set 10, which has much more objects in each run. The results of Set 9 in Table 9 show that the affective learning agent outperformed the active learning agent by an average score of 731 (30.8%). The affective agent outran the active agent by an average of 3,341 milliseconds (12.6%).

Comparing Set 8 results to Set 9 results shows that with the additional attributes, the average score for the affective agent decreased by 893 points, the average score for the active agent decreased by 1,426 points. Both agents scored better in the runs where objects have the fewer attributes. The affective agent showed a smaller decrease in points earned. The difference in the overall score averages between the two agents was 524 showing that the affective agent (smaller difference) was not as affected that much by the increase in attributes as the active reinforcement learning agent. This would indicate that the affective agent is more scalable than the active agent when dealing with more complex objects.

Experiment Set 10 had the following settings: 100 pulses per run , 50,000 objects per run, 10 random attributes per object, with a random reward delay between 3 and 8 pulses. This set increased the number of world objects per pulse from 10,000 to 50,000. This set is to determine how scalable the agent architecture could be.

The results of the experiment are in Table 10. The affective agent outscored the active agent by an average of 461 points (14.5%), and outran the active agent by an average of 4,691 milliseconds (8.05%). Comparing Set 9 to Set 10 shows the effect of

	Affective Agent		Active Agent			
Set 9	Score	Time	Score	Time	Score Diff	Time Diff
211	2,472	23,541	2,423	26,666	49	(3,125)
212	2,713	22,694	2,301	25,282	412	(2,588)
213	3,122	16,513	2,417	26,496	705	(9,983)
214	3,062	23,818	2,100	25,192	962	(1,374)
215	3,203	23,059	2,622	25,560	581	(2,501)
216	3,897	24,127	2,654	27,662	1,243	(3,535)
217	2,961	24,108	2,608	26,635	353	(2,527)
218	3,467	24,237	2,347	27,486	1,120	(3,249)
219	3,262	24,718	2,395	27,192	867	(2,474)
220	2,915	24,022	1,896	26,076	1,019	(2,054)
Avg.	3,107.4	23,083.7	2,376.3	26,424.7	731.1	(3,341.0)
St. Dev	395.23	2,382.67	237.79	884.95	380.24	2,413.42

Table 9: Experiment Set 9: 100 pulses, 10000 objects/pulse, 10 attributes, delay reward

	Affective Agent		Active Agent			
Set 10	Score	Time	Score	Time	Score Diff	Time Diff
171	3,071	66,940	2,523	71,175	548	(4,235)
172	3,731	55,731	3,712	60,538	19	(4,807)
173	3,698	48,948	3,478	59,531	220	(10,583)
174	3,864	54,325	3,669	62,060	195	(7,735)
175	3,383	52,412	3,099	53,675	284	(1,263)
176	3,866	48,732	2,921	53,555	945	(4,823)
177	3,862	47,046	3,041	54,320	821	(7,274)
178	4,087	57,742	3,229	57,814	858	(72)
179	3,517	59,270	3,221	59,776	296	(506)
180	3,420	44,418	2,993	50,031	427	(5,613)
Avg.	3,649.9	53,556.4	3,188.6	58,247.5	461.3	(4,691.1)
St. Dev	300.80	6,719.15	361.15	5,934.24	318.49	3,372.16

Table 10: Experiment Set 10: 100 pulses, 50000 objects/pulse, 10 attributes, delay reward

a larger population of objects for the agent to evaluate. The affective agent still outperforms the active agent significantly, though the active agent's scores increases more when the number of objects increases.

6.7 Experiment Set 11, 12 and 13: 400 Pulses, 50000/50000/100000 #Objects, 10/20/20 #attributes, Reward Waiting

Experiment Set 11 uses the following settings: 400 pulses per run, 50,000 world objects per pulse, 10 random attributes per object, and a reward delay between 3 and 8 pulses. This set extends the run time (number of pulses per run) over Set 9, with all other settings remaining unchanged. This increase in run time shows how well each agent's learning mechanism works with a gigantic input set. The results of the experiments are shown in Table 11. Both agents perform nearly identically on this set. The affective agent outscored the active learning agent by only 1.8%. However, the affective agent processed the inputs 13.3% quicker.

The rewards for individual runs in this set are the most varied of any of the previous sets. In this set, the affective agent does not outperform the active agent in every run. Overall, however, the affective agent's average score is better than the active agent's score. The affective agent does outperform the active agent in terms of completion time in all runs except run 185. Both agents

	Affective Agent		Active Agent			
Set 11	Score	Time	Score	Time	Score Diff	Time Diff
181	16,404	269,595	14,976	286,030	1,428	(16,435)
182	16,267	261,170	16,764	262,030	(497)	(860)
183	16,690	257,344	16,155	281,077	535	(23,733)
184	17,097	223,235	16,750	285,626	347	(62,391)
185	16,998	261,124	15,976	260,534	1,022	590
186	16,893	250,990	15,868	311,886	1,025	(60,896)
187	16,910	186,664	17,060	278,173	(150)	(91,509)
188	16,206	200,823	16,617	246,837	(411)	(46,014)
189	16,866	249,023	17,042	289,857	(176)	(40,834)
190	16,690	236,167	16,885	262,165	(195)	(25,998)
Avg.	16,702.1	239,613.5	16,409.3	276,421.5	292.8	(36,808.0)
St. Dev	311.74	27,783.08	661.21	18,791.40	682.40	29,290.92

Table 11: Experiment Set 11: 400 pulses, 50000 objects/pulse, 10 attributes, delay reward

	Affective Agent		Active Agent			
Set 12	Score	Time	Score	Time	Score Diff	Time Diff
191	29,049	361,929	26,178	442,794	2,871	(80,865)
192	29,858	360,278	25,275	366,706	4,583	(6,428)
193	30,886	346,050	26,269	417,358	4,617	(71,308)
194	30,765	335,677	25,584	375,079	5,181	(39,402)
195	29,343	388,429	29,054	455,908	289	(67,479)
196	30,408	357,357	24,906	389,111	5,502	(31,754)
197	30,126	348,825	25,712	411,338	4,414	(62,513)
198	29,741	306,469	24,535	345,340	5,206	(38,871)
199	30,205	344,477	26,686	362,684	3,519	(18,207)
200	30,434	330,962	26,256	379,692	4,178	(48,730)
Avg.	30,081.5	348,045.3	26,045.5	394,601.0	4,036.0	(46,555.7)
St. Dev	590.65	21,732.39	1,250.38	36,111.13	1,538.16	24,106.60

Table 12: Experiment Set 12: 400 pulses, 50000 objects/pulse, 20 attributes, delay reward

completed in nearly identical time, with the active agent completing 590 milliseconds faster (0.22%) than the affective agent. The increase variability of this set of experiments could be attributed to the decrease in number of attributes. In an environment with fewer values to learn, the order in which the agent is presented the objects could have a greater effect on the results.

Comparing Set 10 to Set 11, increasing the number of runs from 100 to 400 increased the reward by 357% for the affective agent and 414% for the active agent. Both agents experienced a 347% increase in completion time.

Experiment Set 12 has the following settings: 400 pulses per run, 50,000 world objects per run, 20 random attributes per object, with a reward delay between 3 and 8 pulses. The settings of Set 12 are the same as Set 11 except for the increase in the number of attributes. The results of Set 12 are shown in Table 12.

The affective agent performed considerably better against the active agent in this set than in Set 11. The affective agent outscored the active agent by an average of 4,036 points (15.5%) and outperformed the active agent by an average of 46,555 milliseconds (11.8%). The affective agent outperformed the active agent in every run in the set. This difference can be explained by the fact that it is more likely that duplicate objects will be generated with 10 attributes than with 20 attributes. When the agents encounter

	Affective Agent		Active Agent			
Set 13	Score	Time	Score	Time	Score Diff	Time Diff
201	30,194	472,414	25,292	678,866	4,902	(206,452)
202	30,601	570,836	26,648	739,986	3,953	(169,150)
203	31,209	564,484	26,005	712,812	5,204	(148,328)
204	29,956	573,201	26,998	648,448	2,958	(75,247)
205	29,150	553,659	25,949	654,310	3,201	(100,651)
206	17,801	667,135	16,771	903,831	1,030	(236,696)
207	16,710	636,304	15,624	909,233	1,086	(272,929)
208	15,965	628,682	16,769	765,545	(804)	(136,863)
209	16,845	682,812	16,552	888,166	293	(205,354)
210	17,258	649,943	15,215	922,434	2,043	(272,491)
Avg.	23,568.9	599,947.0	21,182.3	782,363.1	2,386.6	(182,416.1)
St. Dev	7,046.10	64,339.06	5,306.66	112,360.06	1,997.82	68,264.25

Table 13: Experiment Set 13: 400 pulses, 100,000 objects/pulse, 20 attributes, delay reward

known objects, the active agent and the affective agent perform about the same. However, when the agents encounter new objects, the affective agent outperformed the active agent.

The final experiment set, Set 13 was run with the following settings: 400 pulses per run, 100,000 world objects per pulse, 20 random attributes per object, and a reward delay between 3 and 8 pulses. The results of this experiment set is shown in Table 13.

In these extremely large sets, the WorldObjectGenerator program had to be rewritten to handle the amount of data generated. Each file in this run set contains 24MB of data. With 400 pulses, the total run for the set exceeded 10GB of data. In comparison, a file with 10,000 objects contained 2MB of data. The run set of 1200 pulses generated 3 GB of data.

The affective agent outscored the active agent by an average score of 2,386 (11.3%). The affective agent outscored the active agent in all runs except one. The affective agent outperformed the active agent in completion time by an average time of 182,416 milliseconds or 182 seconds (23.3%). The active agent took an average of 3 minutes longer to complete than the affective agent. Even in the one run (208) where the active agent outscored the affective agent, the affective agent still completed before the active agent.

6.8 Summary of Experimental Results

All experimental results are summarized in Table 14, affective agent outperforms the active agents in both rewarded earned and completion time, especially when facing large number of objects, each object is associated with large number of attributes. The affective achieved over 10% more reward scores in most (8 out of 12) of the experiment sets. Affective agent also completed all experiments much faster than the active agent, half of the time (6 out of 12) it is over 20% faster.

7 Conclusions & Future Work

In conclusion, the data from these experiments, does support the research hypothesis that an affect-based filtering mechanism can be added to a problem-solving agent architecture in order to decreases the run time without decreasing the agent's utility. The data also indicates that the affect filter has a greater impact on time and performance as the environment becomes more complex. The applications for this research include research into cognitive modeling, autonomous artificial intelligent systems and multi-dimensional reward environments, cognition and problem solving involves emotion. To continue to increase our understanding

Table 14: Summary of Experimental Results

Exp. Set	delay reward	pulse number	objects number	attributes number	Affective Agent over Active agent	
					Reward More	Run Faster
1	No	1	10,000		35.8%	2.3%
2	Yes	300	10,000	15	11%	30%
3	Yes	600	10,000	15	15.2%	20.1%
4	Yes	900	10,000	15	10.5%	23.6%
5	Yes	1200	10,000	15	14.2%	12.2%
6	Yes	300	20,000	15	8.9%	21.1%
7	Yes	600	20,000	15	9.9%	18.0%
8	Yes	100	10,000	15	5.4%	21.5%
9	Yes	100	10,000	10	30.8%	12.6%
10	Yes	100	50,000	10	14.5%	8.05%
11	Yes	400	50,000	10	1.8%	13.3%
12	Yes	400	50,000	20	15.5%	11.8%
13	Yes	400	100,000	20	11.3%	23.3%

of how our own thinking processes work, we will need to continue to research the role that affect plays in that process. Another area where an affect filtering system would be applicable is in the area of multi-autonomous agents. In the development of multi agent systems, each agent needs to be enabled to make decisions on its own. The more complex the environment such agents are placed it, with larger option sets to select from, the more such agents will need some mechanism to distinguish between important and unimportant inputs. In this project, we limited the reward system to a one-dimension utility. Going forward, this design could be used for a multi-dimensional reward, or in situations where the agent would need to decide between different types of rewards rather than just determining the highest value of the reward.

References

[1] A. Abraham, H. Guo, and H. Liu. *Swarm Intelligence: Foundations, Perspectives and Applications*, pages 3–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[2] B. Akhgar, E. Salahi Parvin, and M. H. Sherkat. Axiomatic agent based architecture for agile decision making in strategic information systems. *Journal of Ambient Intelligence and Humanized Computing*, 5(1):93–104, Feb 2014.

[3] S. M. Alarcao and M. J. Fonseca. Emotions recognition using EEG signals: A survey. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[4] E. M. Albornoz and D. H. Milone. Emotion recognition in never-seen languages using a novel ensemble method with emotion profiles. *IEEE Transactions on Affective Computing*, 8(1):43–53, Jan 2017.

[5] J. R. Anderson, M. Matessa, and C. Lebiere. ACT-R: A theory of higher level cognition and its relation to visual attention. *Hum.-Comput. Interact.*, 12(4):439–462, Dec. 1997.

[6] J. Bates, A. B. Loyall, and W. S. Reilly. An architecture for action, emotion, and social behavior. In *Artificial Social Systems: Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Springer-Verlag, Berlin, 1994.

[7] R. F. Baumeister, K. D. Vohs, C. N. DeWall, and L. Zhang. How emotion shapes behavior: Feedback, Anticipation, and Reflection, rather than direct causation. *Personality and Social Psychology Review*, 11(2):167–203, 2007. PMID: 18453461.

[8] R. V. Belavkin. The role of emotion in problem solving. In *Proceedings of the AISB'01 symposium on Emotion, Cognition and Affective Computing*, Heslington, York, England, 2001.

[9] S. Bozinovski and L. Bozinovska. Emotion as internal state evaluation: A connectionist emotion-based learning architecture. In *Emotion-Based Agent Architectures (EBAA 99) - Workshop of the Third International Conference on AUTONOMOUS AGENTS (Agents '99)*, Seattle, USA, May 1999.

[10] E. Cambria, A. Livingstone, and A. Hussain. *The Hourglass of Emotions*, pages 144–157. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[11] A. Chatchinarat, K. W. Wong, and C. C. Fung. Fuzzy classification of human emotions using fuzzy c-mean FCFCM. In *2016 International Conference on Fuzzy Theory and Its Applications (iFuzzy)*, pages 1–5, Nov 2016.

[12] P. Cohen, A. Cheyer, E. Horvitz, R. El Kaliouby, and S. Whittaker. On the future of personal assistants. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1032–1037, New York, NY, USA, 2016. ACM.

[13] A. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Avon Books, 1994.

[14] D. N. Davis. Cognitive architectures for affect and motivation. *Cognitive Computation*, 2(3):199–216, Sep 2010.

[15] de Campos and L. M. Lima. Optimization strategy of neural networks based on rational agents. *International Journal of Hybrid Intelligent Systems*, 14(3):99–112, 2017.

[16] M. Dyer. Emotions and their computations: Three computer models. *Cognition and Emotion*, 1987.

[17] M. S. El-Nasr, T. R. Ioerger, and J. Yen. PETEEI: a PET with evolving emotional intelligence. In *Proceedings of the third annual conference on Autonomous Agents*, pages 9–15. ACM Press, 1999.

[18] M. S. El-Nasr, J. Yen, and T. R. Ioerger. Flame—Fuzzy Logic Adaptive Model of Emotions. *Autonomous Agents and Multi-Agent Systems*, 3(3):219–257, Sep 2000.

[19] C. Elliott. *The Affective Reasoner: A process model of emotions in a multi-agent system.* PhD thesis, Institute for the Learning Sciences Tech., 1992.

[20] H. M. Faisal, M. Ahmad, S. Asghar, and A. Rahman. Select this result for bulk action intelligent quranic story builder. *International Journal of Hybrid Intelligent Systems*, 14(1-2):41–48, 2017.

[21] S. C. Gadanho and P. Dayan. Learning behavior-selection by emotions and cognition in a multi-goal robot task. *Journal of Machine Learning Research*, 4:385–412, 2003.

[22] J. Gratch. Why you should buy an emotional planner. In *Emotion-Based Agent Architectures (EBAA 99) - Workshop of the Third International Conference on Autonomous Agents (Agents '99)*, Seattle, USA, May 1999.

[23] E. Hudlicka. Beyond cognition: Modeling emotion in cognitive architectures. In *Proceedings of the International Conference on Cognitive Modeling (ICCM)*, pages 118–123, 2004.

[24] Y. Kim and A. L. Baylor. Research-based design of pedagogical agent roles: a review, progress, and recommendations. *International Journal of Artificial Intelligence in Education*, 26(1):160–169, Mar 2016.

[25] J. E. Laird. *The Soar cognitive architecture*. 2012.

[26] X. Li, Y. Rao, H. Xie, R. Y. K. Lau, J. Yin, and F. L. Wang. Bootstrapping social emotion classification with semantically rich hybrid neural networks. *IEEE Transactions on Affective Computing*, 8(4):428–442, Oct 2017.

[27] B. Martinez, M. F. Valstar, B. Jiang, and M. Pantic. Automatic analysis of facial actions: A survey. *IEEE Transactions on Affective Computing*, PP(99):1–1, 2017.

[28] B. Meuleman and D. Rudrauf. Induction and profiling of strong multi-componential emotions in virtual reality. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[29] M. Minsky. *The Society of Mind*. Simon & Schuster, Inc., New York, NY, USA, 1986.

[30] O. H. Mowrer. *Learning theory and behavior*. John Wiley & Sons Inc, Hoboken, NJ, US, 1960.

[31] A. C. L. Ngo, J. See, and R. C. . Phan. Sparsity in dynamics of spontaneous subtle emotions: Analysis and application. *IEEE Transactions on Affective Computing*, 8(3):396–411, July 2017.

[32] K. N. Ochsner and J. J. Gross. The cognitive control of emotion. *Trends in Cognitive Sciences*, 9(5):242 – 249, 2005.

[33] A. Ortony, G. Clore, and A. Collins. *The Cognitive Structure of Emotions.* Cambridge University Press, New York, 1988.

[34] T. Pejsa, S. Andrist, M. Gleicher, and B. Mutlu. Gaze and attention management for embodied conversational agents. *ACM Trans. Interact. Intell. Syst.*, 5(1):3:1–3:34, Mar. 2015.

[35] R. Picard. *Affective Computing*. MIT Press, 1997.

[36] M. Poel, R. op den Akker, A. Nijholt, and A.-J. van Kesteren. Learning emotions in virtual environments. In *Agent Construction and Emotions (ACE 2002): A Symposium at the 16th European Meeting on Cybernetics and Systems Research (EMCSR 2002)*, Vienna, Austria, April 2002.

[37] D. K. Pratihar and B. Pratihar. A review on applications of soft computing in design and development of intelligent autonomous robots. *International Journal of Hybrid Intelligent Systems*, 14(1-2):49–65, 2017.

[38] H. Prendinger and M. Ishizuka. Appraisal and filter programs for affective communication. In *AAAI Fall Symposium - Emotional and Intelligent II: The Tangled Knot of Social Cognition*, North Falmouth, Massachusetts, 2-4 November 2001. AAAI Press.

[39] C. Raïevsky and F. Michaud. *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, chapter Emotion Generation Based on a Mismatch Theory of Emotions for Situated Agents, pages 247–266. IGI Global, 2009.

[40] O. O. Rudovic. Machine learning for affective computing and its applications to automated measurement of human facial affect. In *2016 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pages 1–1, Nov 2016.

[41] M. Scheutz and B. Logan. Affective vs. deliberative agent control. In *Proceedings of the AISB'01 symposium on Emotion, Cognition and Affective Computing*, Heslington, York, England, 2001.

[42] K. P. Seng, L. Ang, and C. S. Ooi. A combined rule-based amp;amp; machine learning audio-visual emotion recognition approach. *IEEE Transactions on Affective Computing*, 9(1):3–13, Jan 2018.

[43] R. Subramanian, J. Wache, M. K. Abadi, R. L. Vieriu, S. Winkler, and N. Sebe. ASCERTAIN: Emotion and personality recognition using commercial sensors. *IEEE Transactions on Affective Computing*, 9(2):147–160, April 2018.

[44] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[45] J. D. Velasquez. *When Robots Weep: A Computational Approach to Affective Learning*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Cambridge, MA, USA, 2007.

[46] R. Ventura and C. Pinto-Ferreira. Responding efficiently to relevant stimuli using an emotion-based agent architecture. *Neurocomputing*, 72(13-15):2923–2930, Aug. 2009.

[47] J. R. Wilson and M. Scheutz. A model of empathy to shape trolley problem moral judgements. In *Proceedings of the 2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, ACII '15, pages 112–118, Washington, DC, USA, 2015. IEEE Computer Society.

[48] R. A. Wilson and F. C. Keil, editors. *The MIT Encyclopedia of the Cognitive Sciences (MITECS)*. MIT Press, 2001.

[49] R. Xia and Y. Liu. A multi-task learning framework for emotion recognition using 2d continuous space. *IEEE Transactions on Affective Computing*, 8(1):3–14, Jan 2017.

[50] G. N. Yannakakis, K. Karpouzis, A. Paiva, and E. Hudlicka. Emotion in games. In *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction - Volume Part II*, ACII'11, pages 497–497, Berlin, Heidelberg, 2011. Springer-Verlag.

[51] R. Zhao and K. Mao. Cyberbullying detection based on semantic-enhanced marginalized denoising auto-encoder. *IEEE Transactions on Affective Computing*, 8(3):328–339, July 2017.